Forouzan & Mosharraf
REDES DE
COMPUTADORES
UMA ABORDAGEM TOP-DOWN
Mc
Graw
Hill
bookman

F727r Forouzan, Behrouz A.

Redes de computadores [recurso eletrônico] : uma abordagem top-down / Behrouz A. Forouzan, Firouz Mosharraf ; tradução técnica: Marcos A. Simplicio Jr., Charles Christian Miers. – Dados eletrônicos. – Porto Alegre : AMGH, 2013.

Editado também como livro impresso em 2013.
ISBN 978-85-8055-169-3

1. Computação. 2. Redes de computadores. I. Mosharraf, Firouz. II. Título.

CDU 004.7

Catalogação na publicação: Ana Paula M. Magnus – CRB 10/2052

Behrouz A. Forouzan
De Anza College

Firouz Mosharraf
Rio Hondo College

REDES DE COMPUTADORES

UMA ABORDAGEM TOP-DOWN

Tradução Técnica

Marcos A. Simplicio Jr.

Doutor em Engenharia Elétrica/Sistemas Digitais pela
Escola Politécnica da Universidade de São Paulo
Professor de graduação e pós-graduação nas áreas de Redes de Computadores
e Segurança de Sistemas da Escola Politécnica da Universidade de São Paulo

Charles Christian Miers

Doutor em Engenharia Elétrica/Sistemas Digitais pela
Escola Politécnica da Universidade de São Paulo
Professor de graduação e pós-graduação nas áreas de Ciência da
Computação e Sistemas Operacionais, Redes de Computadores e
Segurança da Informação do Departamento de Ciência da Computação
da Universidade do Estado de Santa Catarina

Versão impressa
desta obra: 2013

AMGH Editora Ltda.
2013

Aos nossos amados:
Ryan, Lily, Melody, Justin e *William.*

APRESENTAÇÃO À EDIÇÃO BRASILEIRA

É cada vez maior a importância das redes de computadores no dia a dia das empresas e das pessoas, de modo geral. A tecnologia deixou de ser apenas um meio de transferência de dados para uso exclusivo das organizações e se tornou um serviço amplamente utilizado nas diversas atividades profissionais e pessoais. Assim, a rápida popularização de redes sem fio, aliada à expansão da malha de fibra óptica, culminou em um ambiente em que a computação é algo tão intrínseco que muitas pessoas sequer percebem o quanto dependem desses serviços.

No Brasil, não poderia ser diferente. Aqui, o mercado de tecnologia da informação cresce rapidamente, com um contínuo aumento no número de provedores de acesso e serviços de Internet de banda larga. Segundo dados do Centro de Estudos sobre as Tecnologias da Informação e da Comunicação (CETIC), embora a universalização do acesso à Internet no Brasil ainda esteja longe de ser uma realidade, observou-se um aumento considerável no número de casas "conectadas" na primeira década deste século. A maioria dos acessos é feita por conexões de banda larga (principalmente com tecnologias como DSL e cabo coaxial), e as velocidades aumentam a cada ano. Ainda há, é claro, muito espaço para a expansão da área de cobertura, especialmente quando comparamos o Brasil aos países que já oferecem, em grande parte de seu território, serviço de *fiber-to-the-home* com taxas de transferência que começam em 100 Mbps e podem ultrapassar 1 Gbps, viabilizando serviços como transmissão de conteúdo em alta definição (HD – High Definition). No entanto, essa discrepância tem levado governos e provedores a buscarem cada vez mais soluções para democratizar serviços de banda larga com maiores taxas de transferência a custos aceitáveis. No Brasil, a expansão da malha de fibra óptica tem evoluído em uma velocidade menor do que a demanda, em especial devido à relação custo-benefício dessa implantação. De fato, muitas vezes, a alternativa adotada tem sido o uso de tecnologias sem fio, que exigem menor investimento em infraestrutura e solucionam parte do problema. Em contrapartida, a adoção de padrões e frequências diferentes dos outros países impõe aos consumidores brasileiros o uso de equipamentos específicos, o que restringe a concorrência e acaba aumentando o seu custo relativo.

Ao mesmo tempo em que o aumento no número de brasileiros com acesso à Internet traz grandes oportunidades para profissionais da área de redes de forma geral, diversos desafios ainda precisam ser superados. Alguns deles estão relacionados à segurança dos dados que trafegam e são armazenados em recursos de rede, como computação em nuvem. Já as questões envolvendo segurança em redes de computadores começaram na década de 1960, e vários problemas do passado já foram resolvidos e muitos outros surgem diariamente devido aos novos serviços ou à descoberta de vulnerabilidades nos serviços já existentes. Embora o foco principal deste livro não seja os mecanismos de segurança, o domínio sobre o conteúdo aqui apresentado é o primeiro passo para quem deseja aprofundar-se na área de segurança de redes de computadores e entender esses problemas.

Além dos aspectos técnicos da segurança, existem também as questões legais para identificar e punir crimes cibernéticos. A diversidade de leis e interpretações do que seria crime em cada país, considerando também a possibilidade de um crime envolver vários países, abre uma margem considerável à impunidade. Entretanto, muitos governos vêm realizando acordos internacionais e aprimorando suas legislações para lidar com esses casos, inclusive o Brasil. Isto é importante, porque problemas de segurança envolvendo redes de computadores são cada vez mais comuns por aqui (por exemplo, ataques de negação de serviço, como aqueles que sobrecarregaram os *sites* de instituições bancárias no início de 2012), conforme reportado pelo Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil (CERT).

Diversas questões técnicas importantes também permeiam o dia a dia dos profissionais de redes; uma delas refere-se ao esgotamento dos endereços IPv4, problema que acelera cada vez mais a implantação da nova geração do IP, o IPv6, no Brasil. Finalmente, a expansão do número de telefones celulares com acesso a serviços de dados, por exemplo, 3G e LTE, tem criado oportunidades para a criação de aplicações avançadas específicas para esses dispositivos.

No domínio da pesquisa, um tema bastante atual refere-se à "Internet do Futuro" e aos elementos que farão parte de sua composição. Questões avançadas, como mobilidade, multimídia, segurança, facilidade de configuração e gerenciamento, são constantes nessas discussões, nas quais o Brasil tem contribuído por meio de diversos projetos de pesquisa internacionais (por exemplo, em iniciativas como o *Framework Programme*, da União Europeia). Além disso, a participação ativa do Brasil em grupos de trabalho internacionais que discutem novos padrões e protocolos para redes de computadores, como IEEE e ITU-T, mostra que os pesquisadores brasileiros, tanto em universidades como em organizações privadas, deixaram de ser espectadores e se tornaram participantes ativos das evoluções da rede.

É nesse ambiente de constante evolução que se baseia o conteúdo aqui abordado. A capacidade de entender os desafios, bem como aproveitar as diversas oportunidades que os acompanham, exige o conhecimento de diversas questões abordadas neste livro, que é uma referência importante tanto para estudantes interessados na área de redes, quanto para os profissionais que nela atuam.

Marcos A. Simplicio Jr.
Doutor em Engenharia Elétrica/Sistemas Digitais pela Escola Politécnica da USP/SP

Charles Christian Miers
Doutor em Engenharia Elétrica/Sistemas Digitais pela Escola Politécnica da USP/SP

PREFÁCIO

As tecnologias relacionadas às redes e à interconexão de redes estão entre aquelas que apresentam um crescimento mais rápido em nossa cultura atualmente. O surgimento de novos aplicativos de redes sociais a cada ano é uma prova dessa afirmação. As pessoas usam a Internet cada vez mais para pesquisas, compras, reservas de passagens aéreas, verificação das últimas notícias, previsão do tempo e assim por diante.

Nesta sociedade orientada à Internet, os especialistas precisam ser treinados para operá-la e gerenciá-la total ou parcialmente ou para cuidar da rede de uma organização que está conectada à Internet. Este livro foi concebido para ajudar estudantes a compreender os conceitos básicos de redes em geral e os protocolos utilizados na Internet em particular.

Características

Embora o objetivo principal do livro seja ensinar os princípios da área de redes, destacamos também outras aplicações:

Protocolos em camadas

O livro foi projetado para ensinar os princípios de redes usando protocolos em camadas da Internet e da pilha de protocolos TCP/IP. Alguns dos princípios de redes podem ter sido duplicados em algumas dessas camadas, porém, em cada caso, eles são apresentados com seus próprios detalhes específicos. Ensinar esses princípios usando protocolos em camadas tem seus benefícios porque tais princípios são repetidos e melhor compreendidos com relação a cada camada. Por exemplo, embora endereçamento seja um problema que se aplique a quatro camadas da pilha de protocolos TCP/IP, cada camada utiliza um formato diferente de endereçamento para finalidades diferentes. Além disso, o endereçamento apresenta um domínio distinto em cada camada. Outro exemplo refere-se ao conceito de enquadramento e empacotamento, repetido em várias camadas, porém cada uma trata o princípio de maneira diferente.

Abordagem *top-down*

Apesar de um dos autores desta obra já ter escrito vários livros sobre redes e Internet (*Comunicação de Dados e Redes de Computadores*, *Protocolo TCP/IP*, *Cryptography and Network Security** e *Local Area Networks**), neste livro a abordagem do tema de redes será diferente: será adotada a abordagem *top-down*.

Embora cada camada da pilha de protocolos TCP/IP dependa dos serviços prestados pela camada abaixo dela, existem duas abordagens para aprender sobre os seus princípios. Na abordagem *bottom-up*, ou de baixo para cima, aprendemos como os *bits* e os sinais se movem na camada física

* N. de T.: Esses livros não têm versão em português. Seus títulos podem ser traduzidos como "Criptografia e Segurança de Redes" e "Redes Locais", respectivamente.

antes de aprendermos como os aplicativos usam esses *bits* para enviar mensagens. Na abordagem *top-down*, ou de cima para baixo, aprendemos primeiro como os protocolos da camada de aplicação trocam mensagens entre si, antes de aprendermos como as mensagens são de fato quebradas em *bits* e sinais e fisicamente transportadas pela Internet.

Público-alvo

Este livro foi escrito tanto para o público acadêmico como para o profissional. Para os profissionais interessados, o livro pode ser utilizado como um guia de autoestudo; para o público acadêmico, como livro-texto, pode ser utilizado para um curso semestral ou quadrimestral. Foi projetado para o último ano de estudos de um curso de graduação ou para o primeiro ano de estudo de um curso de pós-graduação. Embora alguns problemas no final dos capítulos exijam algum conhecimento sobre probabilidades, o estudo do texto requer apenas conhecimentos matemáticos gerais ensinados no primeiro ano de faculdade.

Pedagogia

Diversos recursos pedagógicos deste livro foram concebidos para torná-lo simples de modo que os estudantes compreendam a área de redes de computadores em geral e a Internet em particular.

Abordagem visual

O livro apresenta assuntos altamente técnicos sem utilizar fórmulas complexas, balanceando a quantidade de texto e de figuras. Mais de 670 figuras acompanham o texto, fornecendo uma forma visual e intuitiva de compreender o material. As figuras são especialmente importantes para explicar conceitos de redes que, para muitos estudantes, são mais facilmente compreendidos visualmente do que verbalmente.

Pontos em destaque

Repetimos conceitos importantes usando caixas para rápida referência e atenção imediata.

Exemplos e aplicações

Sempre que apropriado, incluímos exemplos que ilustram os conceitos apresentados no texto. Além disso, adicionamos algumas aplicações reais ao longo de cada capítulo para motivar os estudantes.

Material do final do capítulo

Cada capítulo termina com um conjunto de materiais incluindo os seguintes elementos:

• Termos-chave

Os novos termos usados são listados ao final de cada capítulo e suas definições constam no glossário.

• Resumo

Cada capítulo termina com um resumo que agrupa os assuntos importantes, para que eles sejam vistos em conjunto e de uma só vez.

• Leitura adicional

Seção que fornece uma breve lista de referências relativas ao capítulo, que podem ser usadas para localizar rapidamente a literatura correspondente na seção de referências ao final do livro.

Atividades práticas

Cada capítulo inclui um conjunto de atividades práticas destinadas a reforçar conceitos relevantes e incentivar os estudantes a aplicá-los. Elas são compostas por três partes: testes, questões e problemas.

• Testes*

Os testes, disponíveis no *site* www.mhhe.com/forouzan (em inglês), permitem a rápida verificação dos conceitos. Os estudantes podem fazer esses testes para verificar a sua compreensão dos materiais abordados no capítulo, e o gabarito é fornecido imediatamente.

• Questões

Seção que contém perguntas simples sobre os conceitos discutidos no livro. As respostas, em inglês, às questões ímpares encontram-se disponíveis no *site* www.grupoa.com.br para serem verificadas pelos estudantes.

• Problemas

Seção que contém problemas mais difíceis, que exigem uma compreensão mais profunda dos assuntos discutidos no capítulo. Recomendamos que o estudante tente resolver todos esses problemas. As respostas, em inglês, aos problemas ímpares também encontram-se disponíveis no *site* www.grupoa.com.br para serem verificadas pelos estudantes.

Experimentos de simulação

Conceitos de rede, bem como o fluxo e o conteúdo dos pacotes, podem ser mais bem compreendidos se forem analisados em ação. A maioria dos capítulos inclui uma seção para ajudar os estudantes a realizar experimentos com esses conceitos. Essa seção é dividida em duas partes:

• *Applets**

Applets Java com experimentos interativos foram criados pelos autores e publicados no *site* www.mhhe.com/forouzan, em inglês. Alguns desses *applets* são usados para permitir um melhor entendimento das soluções para alguns problemas, enquanto outros são usados para permitir a melhor compreensão dos conceitos de rede em ação.

• Experimentos de laboratório

Alguns capítulos incluem experimentos de laboratório que usam o Wireshark, um *software* de captura de pacotes. As instruções sobre como obter e usar o Wireshark são fornecidas no Capítulo 1 e o material disponível no *site* do Grupo A está em inglês. Em outros capítulos, existem alguns experimentos de laboratório que podem ser usados para praticar o envio e a recepção de pacotes e para analisar seus conteúdos.

* N. de E.: Os Testes e os *Applets* estão disponíveis apenas no *site* original do livro. Por esse motivo, não nos responsabilizamos por qualquer alteração ou retirada de conteúdo feita pela McGraw-Hill Companies, detentora de todos os direitos.

Tarefas de programação

Alguns capítulos incluem também tarefas de programação. Escrever um programa relativo a um processo ou procedimento esclarece muito as sutilezas e ajuda o estudante a entender melhor o conceito por trás do processo. Embora o estudante possa escrever e testar programas em qualquer linguagem de computador que ele desejar, as soluções, em inglês, fornecidas no *site* do Grupo A para os professores usam a linguagem Java.

Apêndices

Os apêndices, disponíveis no *site* do Grupo A, têm como objetivo fornecer uma referência rápida ou revisão de assuntos necessários para a compreensão dos conceitos discutidos no livro.

Glossário e siglas

O livro contém um extenso glossário e uma lista de siglas que permitem ao leitor encontrar o termo correspondente rapidamente.

Recursos para professores

O livro contém recursos completos para aulas que também podem ser obtidos no *site* www.grupoa.com.br. Eles incluem:

Apresentações e Biblioteca de imagens

Um conjunto de apresentações em PowerPoint, em português, com cores e animações para auxiliar no ensino de disciplinas de redes.

Respostas para as atividades práticas

Respostas em inglês, para todas as questões e para todos os problemas são fornecidas no *site* para uso dos professores que lecionam disciplinas de redes.

Respostas para as tarefas de programação

Respostas em inglês, para as tarefas de programação também estão disponíveis no *site*. Os programas foram escritos na linguagem C para o Capítulo 2 e na linguagem Java para os demais capítulos.

Como usar o livro

Os capítulos do livro foram organizados com o objetivo de proporcionar uma grande flexibilidade na sua leitura. Sugerimos o seguinte:

- A maioria do material discutido no Capítulo 1 é essencial para a compreensão do restante do livro. As duas primeiras seções são fundamentais para compreender o conceito de protocolos em camadas sobre o qual todo o livro foi concebido. As duas últimas intituladas "História da Internet" e "Padrões e administração", podem ser ignoradas ou consideradas material de autoestudo.
- Os Capítulos 2 a 6 abordam as quatro camadas mais altas da pilha de protocolos TCP/IP. Sugerimos que eles sejam abordados na ordem apresentada para preservar a abordagem *top-down* adotada no livro. Entretanto, existem algumas seções, como "Progra-

mação usando a interface *socket*" no Capítulo 2, "Nova geração do IP" no Capítulo 4, ou "Outras redes com fios" no Capítulo 5, que podem ser ignoradas sem afetar a continuidade do livro.

- O Capítulo 7, intitulado "Camada física", foi adicionado ao livro para tornar a discussão da pilha de protocolos TCP/IP mais completa. Ele pode ser ignorado se o professor considerar que os estudantes já estão familiarizados com o assunto ou que o tenham aprendido em alguma outra disciplina relacionada.
- Os Capítulos 8, 9 e 10 podem ser ensinados em qualquer ordem após a discussão dos seis primeiros capítulos. Podem ser ensinados total ou parcialmente, ou mesmo totalmente ignorados, a critério do professor.
- O Capítulo 11 é dedicado à programação em Java voltada a redes. Ele tem duas finalidades: primeiro, apresenta a ideia de programação cliente-servidor para permitir que os estudantes entendam melhor o propósito da Internet. Segundo, prepara o estudante para disciplinas mais avançadas sobre programação para redes. Uma miniduplicata desse capítulo é apresentada no Capítulo 2, usando a linguagem C. O professor pode usar tanto essa seção como o Capítulo 11 para ensinar os conceitos básicos de programação para redes.

Marcas registradas

Ao longo do texto, utilizamos diversas marcas registradas. Em vez de inserir o símbolo correspondente (®) a cada menção ao nome das marcas registradas, as reconhecemos aqui e afirmamos que elas são usadas sem qualquer intenção de infringi-las. Outros nomes de produtos, marcas comerciais e registradas são de propriedade de seus respectivos donos.

Agradecimentos

É óbvio que o desenvolvimento de um livro desse âmbito requer o apoio de muitas pessoas. Gostaríamos de agradecer as contribuições de alguns revisores. Eles são:

Zongming Fei	University of Kentucky
Randy J. Fortier	University of Windsor
Seyed H. Hosseini	University of Wisconsin, Milwaukee
George Kesidis	Pennsylvania State University
Amin Vahdat	University of California, San Diego
Yannis Viniotis	North Carolina State University
Bin Wang	Wright State University
Vincent Wong	University of British Columbia
Zhi-Li Zhang	University of Minnesota
Wenbing Zhao	Cleveland State University

Agradecemos especialmente à equipe da McGraw-Hill. Raghu Srinivasan, o editor, provou que um editor proficiente pode transformar o impossível em possível. Melinda Bilecki, a editora de desenvolvimento, nos ajudou sempre que precisamos. Jane Mohr, gerente de projeto, nos guiou ao longo do processo de produção com enorme entusiasmo. Também agradecemos a Dheeraj Chahal, gerente de projeto, Brenda A. Rolwes, projetista da capa, e Kathryn DiBernardo, editora de texto.

Forouzan e Mosharraf
Los Angeles, CA.

SIGLAS

2B1Q	*Two binary, one quaternary*	2-binário, 1-quaternário
4B/5B	*Four binary/five binary*	4-binário, 5 binário
4D-PAM5	*4-Dimensional, 5-Level Pulse Amplitude Modulation*	Modulação de Amplitude de Pulso de 5 Níveis e 4 Dimensões
8B/10B	*Eight binary/ten binary*	8 binário, 10 binário
8B6T	*Eight binary, six ternary*	8 binário, 6 ternário
AAL	*Application Adaptation Layer*	Camada de Adaptação ATM
AAS	*Adaptive Antenna System*	Sistema de Antenas Adaptativas
ABM	*Asynchronous Balanced Mode*	Modo Balanceado Assíncrono
ABR	*Available Bit Rate*	Taxa de *Bits* Disponível
AC	*certification authority*	Autoridade Certificadora
ACK	*acknowledgment*	confirmação
ACL	*Asynchronous Connectionless Link*	Enlace Assíncrono Sem Conexão
ADM	*Adaptive DM*	DM Adaptativo
ADPCM	*Adaptive DM*	DPCM Adaptativo
ADSL	*Asymmetric Digital Subscriber Line*	Linha de Assinante Digital Assimétrica
AES	*Advanced Encryption Standard*	Padrão Avançado de Cifração
AH	*Authentication Header*	Cabeçalho de Autenticação
AIMD	*Additive Increase, Multiplicative Decrease*	Aumento Aditivo, Diminuição Multiplicativa
AM	*Amplitude Modulation*	Modulação de Amplitude
AMI	*Alternate Mark Inversion*	Inversão Alternada de Marcas
AMPS	*Advanced Mobile Phone System*	Sistema Avançado de Telefonia Móvel
ANSI	*American National Standards Institute*	Instituto Nacional Americano de Padrões
ANSNET	*Advanced Networks and Services Network*	Rede ANS
AP	*Access Point*	Ponto de Acesso
API	*Application Programming Interface*	Interface de Programação de Aplicativos
APS	*Automatic Protection Switching*	Proteção por Comutação Automática
ARP	*Address Resolution Protocol*	Protocolo de Resolução de Endereços
ARPA	*Advanced Research Projects Agency*	Agência de Projetos de Pesquisa Avançados
ARPANET	*Advanced Research Projects Agency Network*	Rede ARPA
ARQ	*Automatic Repeat reQuest*	Solicitação de Repetição Automática
AS	*Authentication Server*	Servidor de Autenticação
AS	*Autonomous System*	Sistema Autônomo
ASCII	*American Standard Code for Information Interchange*	Código Padrão Americano para o Intercâmbio de Informações

ASK	*Amplitude Shift Keying*	Modulação por Chaveamento de Amplitude
ASN.1	*Abstract Syntax Notation One*	Notação Sintática Abstrata Um
ATM	*Asynchronous Transfer Mode*	Modo de Transferência Assíncrona
AUI	*Attachment Unit Interface*	Interface com Unidade para Anexação
B8ZS	*Bipolar with 8-Zero Substitution*	Bipolar com Substituição de 8 Zeros
Bc	*committed burst size*	tamanho de rajada autorizado
BECN	*Backward Explicit Congestion Notification*	Notificação Explícita de Congestionamento no Sentido Reverso
BER	*Basic Encoding Rules*	Regras Básicas de Codificação
BGP	*Border Gateway Protocol*	Protocolo de Roteamento de Borda
BNC	*Bayone-Neill-Concelman*	Conector de Cabo Coaxial Comum
BOOTP	*Bootstrap Protocol*	Protocolo de Inicialização
BRI	*Basic Rate Interface*	Interface de Taxa Básica
BSS	*Basic Service Set*	Conjunto Básico de Serviços
CATV	*Community Antenna TV*	Antena Comunitária de TV
CBC	*Cipher-Block Chaining*	Encadeamento de Blocos
CBR	*Constant Bit Rate*	Taxa Constante de *Bits*
CBT	*Core-Based Tree*	Árvore Baseada em Núcleo
CC	—	Corrente Contínua
CCITT	*Consultative Committee for International Telegraphy and Telephony*	Comitê Consultivo Internacional de Telegrafia e Telefonia
CCK	*Complementary Code Keying*	Chaveamento de Código Complementar
CDMA	*Code Division Multiple Access*	Acesso Múltiplo por Divisão de Código
CDPD	*Cellular Digital Packet Data*	Pacotes de Dados Digitais via Rede Celular
CDV	*Cell Delay Variation*	Variação de Atraso da Célula
CEPT	*Comité Européen de Post et Telegraphie*	Comitê Europeu de Correios e Telégrafos
CGI	*Common Gateway Interface*	Interface de Comunicação Comum
CHAP	*Challenge Handshake Authentication Protocol*	Protocolo de Autenticação por Desafio-Resposta
CIDR	*Classless Interdomain Routing*	Roteamento Interdomínios sem Classes
CIR	*Committed Information Rate*	Taxa de Informações Autorizada
CLP	*Cell Loss Priority*	Prioridade de Perda de Células
CLR	*Cell Loss Ratio*	Taxa de Perda de Células
CMS	*Cryptographic Message Syntax*	Sintaxe para Mensagens Cifradas
CMTS	*Cable Modem Transmission System*	Sistema de Transmissão por Modem a Cabo
CPE	*Customer Premises Equipment*	Equipamento nas Instalações do Cliente
CRC	*Cyclic Redundancy Check*	Verificação de Redundância Cíclica
CS	*Convergence Sublayer*	Subcamada de Convergência
CSM	*Cipher Stream Mode*	Modo de Cifração em Fluxo
CSMA	*Carrier Sense Multiple Access*	Acesso Múltiplo com Detecção de Portadora
CSMA/CA	*Carrier Sense Multiple Access with Collision Avoidance*	Acesso Múltiplo com Detecção de Portadora / Prevenção de Colisão
CSMA/CD	*Carrier Sense Multiple Access with Collision Detection*	Acesso Múltiplo com Detecção de Portadora / Detecção de Colisão
CSNET	*Computer Science Network*	Rede de Ciência da Computação

CSRC	*contributing source*	fonte contribuinte
CSS	*Cascading Style Sheets*	Folhas de Estilo em Cascata
CTS	*Clear To Send*	Liberado para Enviar
D-AMPS	*digital AMPS*	AMPS digital
DARPA	*Defense Advanced Research Projects Agency*	Agência de Projetos de Pesquisa Avançados de Defesa
dB	*decibel*	decibel
DCF	*Distributed Coordination Function*	Função de Coordenação Distribuída
DCT	*Discrete Cosine Transform*	Transformada Discreta do Cosseno
DDNS	*Dynamic Domain Name System*	Sistema de Nomes de Domínio Dinâmico
DDS	*Digital Data Service*	Serviço de Dados Digitais
DE	*Discard Eligibility*	Elegibilidade para Descarte
DEMUX	*demultiplexer*	demultiplexador
DES	*Data Encryption Standard*	Padrão de Cifração de Dados
DHCP	*Dynamic Host Configuration Protocol*	Protocolo de Configuração Dinâmica de Host
DHT	*Distributed Hash Table*	Tabela de Hash Distribuída
DiffServ	*Differentiated Services*	Serviços Diferenciados
DIFS	*DCF Interframe Space*	Espaço entre Quadros DCF
DISC	*disconnect*	desconectar
DMT	*Discrete Multitone Technique*	Técnica de Múltiplos Tons Discretos
DNS	*Domain Name System*	Sistema de Nomes de Domínio
DOCSIS	*Data Over Cable System Interface Specification*	Especificação de Interface de Sistema de Dados Sobre Cabo
DPCM	*differential PCM*	PCM diferencial
DS-*n*	*Digital Signal-n*	Sinal Digital-*n*
DS	*Differentiated Services*	Serviços Diferenciados
DSL	*Digital Subscriber Line*	Linha de Assinante Digital
DSLAM	*Digital Subscriber Line Access Multiplexer*	Multiplexador de Acesso a Linha de Assinante Digital
DSS	*Digital Signature Standard*	Padrão de Assinatura Digital
DSSS	*Direct Sequence Spread Spectrum*	Espalhamento Espectral por Sequência Direta
DTE	*Data Terminal Equipment*	Equipamento de Terminal de Dados
DVMRP	*Distance Vector Multicast Routing Protocol*	Protocolo de Roteamento *Multicast* por Vetor de Distâncias
DWDM	*Dense Wave-Division Multiplexing*	Multiplexação por Divisão de Onda Densa
EBCDIC	*Extended Binary Coded Decimal Interchange Code*	Código Estendido de Intercâmbio Decimal Codificado em Binário
eBGP	*external BGP*	BGP externo
ECB	*Electronic CodeBook*	Livro de Códigos Eletrônico
EGP	*Exterior Gateway Protocol*	–
EIA	*Electronic Industries Alliance*	Aliança de Indústrias Eletrônicas
EM	*mobile station*	estação móvel
ENQ	*enquiry frame*	quadro de consulta
ESP	*Encapsulating Security Payload*	Encapsulamento de Dados de Segurança

ESS	*Extended Service Set*	Conjunto Estendido de Serviços
FA	*foreign agent*	agente externo
FCC	*Federal Communications Commission*	Comissão Federal de Comunicações
FCS	*Frame Check Sequence*	Sequência de Verificação de Quadro
FDD	*Frequency Division Duplex*	Duplexação por Divisão de Frequência
FDDI	*Fiber Distributed Data Interface*	Interface de Dados Distribuídos por Fibra
FDM	*Frequency-Division Multiplexing*	Multiplexação por Divisão de Frequência
FDMA	*Frequency Division Multiple Access*	Acesso Múltiplo por Divisão de Frequência
FEC	*Forward Error Correction*	Correção Antecipada de Erros
FHSS	*Frequency Hopping Spread Spectrum*	Espalhamento Espectral por Saltos em Frequências
FIFO	*First-In, First-Out*	Primeiro a Entrar, Primeiro a Sair
FM	*Frequency Modulation*	Modulação de Frequência
FRMR	*frame reject*	rejeição de quadro
FQDN	*Fully Qualified Domain Name*	Nome de Domínio Totalmente Qualificado
FSK	*Frequency Shift Keying*	Modulação por Chaveamento de Frequência
FTP	*File Transfer Protocol*	Protocolo de Transferência de Arquivos
GEO	*Geostationary Earth Orbit*	Órbita Terrestre Geoestacionária
GIF	*Graphical Interchange Format*	Formato para Troca de Gráficos
GPS	*Global Positioning System*	Sistema de Posicionamento Global
GSM	*Global System for Mobile Communication*	Sistema Global para Comunicações Móveis
HA	*home agent*	agente nativo
HDLC	*High-level Data Link Control*	Controle de Enlace de Dados de Alto Nível
HDSL	*High Bit Rate Digital Subscriber Line*	Linha de Assinante Digital de Alta Taxa de Transferência
HEC	*Header Error Check*	Correção de Erros do Cabeçalho
HFC	*Hybrid-Fiber-Coaxial*	Híbrida Fibra-Coaxial
HMAC	*Hashed Message Authentication Code*	Código de Autenticação de Mensagem baseado em Hash
HR-DSSS	*High-Rate Direct-Sequence Spread Spectrum*	Espalhamento Espectral por Sequência Direta de Alta Taxa
HTML	*HyperText Markup Language*	Linguagem de Marcação de Hipertexto
HTTP	*HyperText Transfer Protocol*	Protocolo de Transferência de Hipertexto
Hz	–	hertz
IAB	*Internet Architecture Board*	Conselho de Arquitetura da Internet
IANA	*Internet Assigned Numbers Authority*	Autoridade para Atribuição de Números na Internet
iBGP	*internal BGP*	BGP interno
ICANN	*Internet Corporation for Assigned Names and Numbers*	Corporação para Atribuição de Nomes e Números na Internet
ICMP	*Internet Control Message Protocol*	Protocolo de Mensagens de Controle da Internet
IEEE	*Institute of Electrical and Electronics Engineers*	Instituto de Engenheiros Eletricistas e Eletrônicos
IESG	*Internet Engineering Steering Group*	–
IETF	*Internet Engineering Task Force*	Força-Tarefa de Engenharia da Internet

IFDMA	*interleaved FDMA*	FDMA Intercalado
IFS	*Interframe Space*	Espaço entre Quadros
IGMP	*Internet Group Management Protocol*	Protocolo de Gerenciamento de Grupos Internet
IGP	*Interior Gateway Protocol*	–
IKE	*Internet Key Exchange*	Troca de Chaves da Internet
ILEC	*Incumbent Local Exchange Carrier*	Operadora Local Incumbente
IMAP	*Internet Mail Access Protocol*	Protocolo de Acesso a Correio da Internet
INTERNIC	*Internet Network Information Center*	Núcleo de Informação e Coordenação da Internet
IntServ	*Integrated Services*	Serviços Integrados
IP	*Internet Protocol*	Protocolo Internet
IPCP	*Internetwork Protocol Control Protocol*	Protocolo de Controle de Rede da Internet
IPng	*Internet Protocol, next generation*	nova geração do Protocolo Internet
IPSec	*IP Security*	Segurança IP
IPv6	*Internet Protocol, version 6*	Protocolo Internet, versão 6
IRTF	*Internet Research Task Force*	Força-Tarefa de Pesquisa da Internet
IS-95	*Interim Standard 95*	Padrão Provisório 95
ISAKMP	*Internet Security Association and Key Management Protocol*	Protocolo de Gerenciamento de Chaves e Associações de Segurança da Internet
ISDN	*Integrated Services Digital Network*	Rede Digital de Serviços Integrados
ISN	*Initial Sequence Number*	Número de Sequência Inicial
ISO	*International Organization for Standardization*	Organização Internacional de Padronização
ISOC	*Internet Society*	Sociedade Internet
ISP	*Internet Service Provider*	Provedor de Serviços de Internet
IMT-2000	*Internet Mobile Communication 2000*	Comunicação para Internet Móvel 2000
ITU	*International Telecommunications Union*	União Internacional de Telecomunicações
ITU-T	*ITU, Telecommunication Standardization Sector*	ITU, Setor de Padronização de Telecomunicações
IV	*Initial Vector*	Vetor de Inicialização
JPEG	*Joint Photographic Experts Group*	Grupo de Especialistas em Fotografia
KDC	*Key-Distribution Center*	Centro de Distribuição de Chaves
L2CAP	*Logical Link Control and Adaptation Protocol*	Protocolo de Adaptação e Controle de Enlace Lógico
LAN	*Local Area Network*	Rede Local
LAP	*Line Access Procedure*	Procedimento de Acesso à Linha
LCP	*Link Control Protocol*	Protocolo de Controle de Enlace
LEO	*Low-Earth-Orbit*	Órbita Terrestre Baixa
LIS	*Logical IP Subnet*	Sub-rede IP Lógica
LLC	*Logical Link Control*	Controle de Enlace Lógico
LMI	*Local Management Information*	Informação de Gerenciamento Local
LMP	*Link Management Protocol*	Protocolo de Gerenciamento de Enlace
LPC	*Linear Predictive Coding*	Codificação Preditiva Linear
LSA	*Link-State Advertisement*	Anúncio de Estado de Enlace
LSP	*link-state packet*	pacote de estado de enlace

MA	*Multiple Access*	Acesso Múltiplo
MAA	*Message Access Agent*	Agente de Acesso a Mensagens
MAC	*Media Access Control*	Controle de Acesso ao Meio
MAC	*Message Authentication Code*	Código de Autenticação de Mensagens
MAN	*Metropolitan Area Network*	Rede Metropolitana
MBONE	*multicast backbone*	espinha dorsal *multicast*
MBS	*Maximum Burst Size*	Tamanho Máximo da Rajada
MC-CDMA	*Multi-Carrier CDMA*	CDMA Multiportadora
MD	*Message Digest*	Resumo Criptográfico da Mensagem
MDC	*Modification Detection Code*	Código de Detecção de Modificação
MEO	*Medium-Earth-Orbit*	Órbita Terrestre Média
MIB	*Management Information Base*	Base de Informações de Gerenciamento
MID	*Message Identifier*	Identificador de Mensagem
MII	*Medium Independent Interface*	Interface Independente de Meio
MILNET	*Military Network*	Rede Militar
MIME	*Multipurpose Internet Mail Extensions*	Extensões Multifunção para Mensagens de Internet
MIMO	*Multiple-Input, Multiple-Output antenna*	antena de Múltiplas Entradas e Múltiplas Saídas
MLT-3	*multiline transmission, 3-level*	transmissão multilinha de 3 níveis
modem	—	modulador-demodulador
MOSPF	*Multicast Open Shortest Path First*	Protocolo *Multicast* Aberto de Menor Rota Primeiro
MP3	*MPEG Audio Layer 3*	Áudio MPEG Camada 3
MPEG	*Motion Picture Experts Group*	Grupo de Especialistas em Imagens em Movimento
MPLS	*Multi-Protocol Label Switching*	Comutação de Rótulos Multiprotocolo
MSC	*Mobile Switching Center*	Central de Comutação Móvel
MSS	*Maximum Segment Size*	Tamanho Máximo de Segmento
MTA	*Mail Transfer Agent*	Agente de Transferência de Mensagens
MTSO	*Mobile Telephone Switching Office*	Central de Comutação de Telefonia Móvel
MTU	*Maximum Transfer Unit*	Unidade Máxima de Transferência
MUX	—	multiplexador
NAK	*negative acknowledgment*	confirmação negativa
NAP	*Network Access Point*	Ponto de Acesso à Rede
NAT	*Network Address Translation*	Tradução de Endereços de Rede
NAV	*Network Allocation Vector*	Vetor de Alocação de Rede
NCP	*Network Control Protocol*	Protocolo de Controle de Rede
NIC	*Network Information Center*	Núcleo de Informação e Coordenação
NIC	*Network Interface Card*	Placa de Interface de Rede
NIST	*National Institute of Standards and Technology*	Instituto Nacional de Padrões e Tecnologia
NNI	*Network-to-Network Interface*	Interface Rede-Rede
NRM	*Normal Response Mode*	Modo de Resposta Normal
NRZ	*Non-Return-to-Zero*	Não Retorno ao Zero
NRZ-I	*Non-Return-to-Zero, Invert*	Não Retorno ao Zero-Inversão

NRZ-L	*Non-Return-to-Zero, Level*	Não Retorno ao Zero-Nível
NSA	*National Security Agency*	Agência de Segurança Nacional
NSF	*National Science Foundation*	Fundação Nacional de Ciência
NSFNET	*National Science Foundation Network*	Rede da Fundação Nacional de Ciência
NVT	*Network Virtual Terminal*	Terminal Virtual de Rede
OADM	*Optical Add-Drop Multiplexer*	Multiplexador de Inserção/Remoção Óptico
OC	*Optical Carrier*	Portadora Óptica
OFB	*Output Feedback*	Realimentação de Saída
OFDM	*Orthogonal-Frequency-Division-Multiplexing*	Multiplexação por Divisão de Frequência Ortogonal
OSI	*Open Systems Interconnection*	Interconexão de Sistemas Abertos
OSPF	*Open Shortest Path First*	Protocolo Aberto de Menor Rota Primeiro
P/F	*poll/final*	varredura/final
P2P	*peer-to-peer*	par a par
PAM	*Pulse Amplitude Modulation*	Modulação de Amplitude de Pulso
PAP	*Password Authentication Protocol*	Protocolo de Autenticação por Senha
PC	*Predictive Coding*	Codificação Preditiva
PCF	*Point Coordination Function*	Função de Coordenação Pontual
PCM	*Pulse Code Modulation*	Modulação por Código de Pulsos
PCS	*Personal Communication System*	Sistema de Comunicação Pessoal
PDU	*Protocol Data Unit*	Unidade de Dados de Protocolo
PGP	*Pretty Good Privacy*	Privacidade Bastante Boa
PHB	*per-hop behavior*	comportamento por salto
PIM	*Protocol Independent Multicast*	*Multicast* Independente de Protocolo
PIM-DM	*Protocol Independent Multicast-Dense Mode*	*Multicast* Independente de Protocolo-Modo Denso
PIM-SM	*Protocol Independent Multicast-Sparse Mode*	*Multicast* Independente de Protocolo-Modo Esparso
PING	*Packet Internet Groper*	Procurador de Pacotes na Internet
PKI	*Public Key Infrastructure*	Infraestrutura de Chaves Públicas
PM	*Phase Modulation*	Modulação de Fase
PNNI	*Private Network-to-Network Interface*	Interface Rede-Rede Privada
POP	*Point of Presence*	Ponto de Presença
POP3	*Post Office Protocol, version 3*	Protocolo de Agência de Correio, versão 3
POS	*packet over SONET*	pacote sobre SONET
POTS	*Plain Old Telephone System*	Rede de Telefonia Convencional
PPM	*Pulse Position Modulation*	Modulação por Posição de Pulso
PPP	*Point-to-Point Protocol*	Protocolo Ponto a Ponto
PQDN	*Partially Qualified Domain Name*	Nome de Domínio Parcialmente Qualificado
PS	*Simple Protocol*	Protocolo Simples
PSK	*Phase Shift Keying*	Modulação por Chaveamento de Fase
PSTN	*Public Switched Telephone Network*	Rede de Telefonia Pública Comutada
PVC	*Permanent Virtual Circuit*	Circuito Virtual Permanente
QAM	*Quadrature Amplitude Modulation*	Modulação por Amplitude em Quadratura
QoS	*Quality of Service*	Qualidade de Serviço

Quadro B	*B-frame*	Quadro bidirecional
Quadro I	*I-frame*	Quadro intracodificado
RACE	*Research in Advance Communication for Europe*	Pesquisa em Comunicação Avançada para a Europa
RADSL	*Rate Adaptive Asymmetrical Digital Subscriber Line*	Linha de Assinante Digital Assimétrica com Taxa Adaptativa
RARP	*Reverse Address Resolution Protocol*	Protocolo de Resolução de Endereços Reverso
REJ	*Rereject*	rejeição
RFC	*Request for Comment*	Pedido de Comentários
RIP	*Routing Information Protocol*	Protocolo de Informações de Roteamento
rlogin	*remote login*	acesso remoto
RNR	*Receive Not Ready*	Receptor Não Pronto
ROM	*Read-Only Memory*	Memória Apenas de Leitura
RP	*Rendezvous Point*	Ponto de Encontro
RP	*pseudorandom noise*	ruído pseudoaleatório
RPB	*Reverse Path Broadcasting*	*Broadcast* pelo Caminho Reverso
RPF	*Reverse Path Forwarding*	Encaminhamento pelo Caminho Reverso
RPM	*Reverse Path Multicasting*	*Multicast* pelo Caminho Reverso
RS	*selective repeat*	Repetição Seletiva
RSA	*Rivest, Shamir, Adleman*	Rivest, Shamir, Adleman
RSVP	*Resource Reservation Protocol*	Protocolo de Reserva de Recursos
RTCP	*Real-time Transport Control Protocol*	Protocolo de Controle de Transporte em Tempo Real
RTO	*Retransmission Time-Out*	Tempo Limite de Retransmissão
RTP	*Real-time Transport Protocol*	Protocolo de Transporte em Tempo Real
RTS	*request to send*	solicitação de envio
RTSP	*Real-Time Streaming Protocol*	Protocolo de Fluxo Contínuo em Tempo Real
RTT	*Round-Trip Time*	Tempo de Ida e Volta
RZ	*Return-to-Zero*	Retorno ao Zero
S/MIME	*Secure/Multipurpose Internet Mail Extensions*	Extensões Multifunção para Mensagens de Internet com Segurança
S/R	*Signal-to-Noise Ratio*	Relação Sinal/Ruído
SA	*Security Association*	Associação de Segurança
SAD	*Security Association Database*	Base de Dados de Associações de Segurança
SAR	*Segmentation and Reassembly*	Segmentação e Remontagem
SCCP	*Signaling Connection Control Part*	Parte de Controle de Sinalização de Conexão
SCO	*Synchronous Connection-Oriented*	Enlace Síncrono Orientado à Conexão
SCP	*Server Control Point*	Ponto de Controle de Servidor
SCTP	*Stream Control Transmission Protocol*	Protocolo de Controle de Fluxo de Transmissão
SDH	*Synchronous Digital Hierarchy*	Hierarquia Digital Síncrona
SDR	*Software Defined Radio*	Rádio Definido por Software
SDSL	*Symmetric Digital Subscriber Line*	Linha de Assinante Digital Simétrica
SDU	*Service Data Unit*	Unidade de Dados de Serviço

SEAL	*Simple and Efficient Adaptation Layer*	Camada de Adaptação Simples e Eficiente
SFD	*Start Frame Delimiter*	Delimitador de Início do Quadro
SHA	*Secure Hash Algorithm*	Algoritmo de *Hash* Seguro
SIFS	*Short Interframe Space*	Espaço Curto entre Quadros
SIP	*Session Initiation Protocol*	Protocolo de Iniciação de Sessão
SKEME	*Secure Key Exchange Mechanism*	Mecanismo Seguro de Troca de Chaves
SMI	*Structure of Management Information*	Estrutura de Gerenciamento da Informação
SMTP	*Simple Mail Transfer Protocol*	Protocolo Simples de Transferência de Correio
SNMP	*Simple Network Management Protocol*	Protocolo Simples de Gerenciamento de Rede
SNR	*Signal-to-Noise Ratio*	Relação Sinal/Ruído
SOFDMA	*Scalable OFDMA*	OFDMA Escalável
SONET	*Synchronous Optical Network*	Rede Óptica Síncrona
SP	*Security Policy*	Política de Segurança
SPD	*Security Policy Database*	Base de Dados de Políticas de Segurança
SPE	*Synchronous Payload Envelope*	Envelope de Carga Útil Síncrono
SPI	*Security Parameter Index*	Índice do Parâmetro de Segurança
SREJ		rejeição seletiva
SS	*Spread Spectrum*	Espalhamento Espectral
SS7	*Signaling System Seven*	Sistema de Sinalização 7
SSCS	*Service Specific Convergence sublayer*	subcamada de Convergência de Serviço Específico
SSH	*Secure Shell*	–
SSL	*Secure Sockets Layer*	–
SSN	*stream sequence number*	número de sequência do fluxo
SSRC	*synchronization source*	fonte de sincronização
STM	*Synchronous Transport Module*	Módulo de Transporte Síncrono
STP	*Shielded Twisted-Pair*	Par Trançado Blindado
STS	*Synchronous Transport Signal*	Sinal de Transporte Síncrono
SVC	*Switched Virtual Circuit*	Circuito Virtual Comutado
TCAP	*Transaction Capabilities Application Port*	Porta de Aplicação de Capacidades da Transação
TCB	*Transmission Control Block*	Bloco de Controle da Transmissão
TCP	*Transmission Control Protocol*	Protocolo de Controle de Transmissão
TCP/IP	*Transmission Control Protocol/Internet Protocol*	Protocolo de Controle de Transmissão/ Protocolo Internet
TDD	*Time-Division Duplex*	Duplexação por Divisão de Tempo
TDM	*Time-Division Multiplexing*	Multiplexação por Divisão de Tempo
TDMA	*Time-Division Multiple Access*	Acesso Múltiplo por Divisão de Tempo
TELNET	*Terminal Network*	Terminal de Rede
TFTP	*Trivial File Transfer Protocol*	Protocolo de Transferência de Arquivos Trivial
TLI	*Transport-Layer Interface*	Interface de Camada de Transporte
TLS	*Transport Layer Security*	Protocolo de Segurança da Camada de Transporte

TOS	*Type of Service*	Tipo de Serviço
TP	*Transmission Path*	Caminho de Transmissão
TRPB	*Truncated Reverse-Path Broadcasting*	*Broadcast* Truncado pelo Caminho Reverso
TSI	*Time-Slot Interchange*	Troca por Parcela Discreta de Tempo
TSN	*Transmission Sequence Number*	Número de Sequência da Transmissão
TTL	*Time-To-Live*	Tempo de Vida
TUP	*Telephone User Port*	Porta de Usuário de Telefonia
UA	*User Agent*	Agente de Usuário
UBR	*Unspecified Bit Rate*	Taxa de *Bits* Não Especificada
UDP	*User Datagram Protocol*	Protocolo de Datagramas de Usuário
UMTS	*Universal Mobile Telecommunication System*	Sistema de Telecomunicações Móveis Universal
UNI	*User-to-Network Interface*	Interface Usuário-Rede
URL	*Uniform Resource Locator*	Localizador Uniforme de Recursos
UTP	*Unshielded Twisted-Pair*	Par Trançado Sem Blindagem
VBR	*Variable Bit Rate*	Taxa Variável de *Bits*
VC	*Virtual Circuit*	Circuito Virtual
VCC	*Virtual Circuit Connection*	Conexão de Circuito Virtual
VCI	*Virtual Circuit Identifier*	Identificador de Circuito Virtual
VDSL	*Very High Bit Rate Digital Subscriber Line*	Linha de Assinante Digital de Taxa de Dados Muito Alta
VLAN	*Virtual Local Area Network*	Rede Local Virtual
VOIP	*Voice over IP*	Voz sobre IP
VP	*Virtual Path*	Caminho Virtual
VPI	*Virtual Path Identifier*	Identificador de Caminho Virtual
VPN	*Virtual Private Network*	Rede Virtual Privada
VT	*Virtual Tributary*	Tributários Virtuais
WAN	*wide area network*	rede de longa distância
WDM	*Wavelength-Division Multiplexing*	Multiplexação por Divisão de Comprimento de Onda
WiMAX	*Worldwide Interoperability for Microwave Access*	Interoperabilidade Mundial para Acesso por Micro-ondas
WWW	*World Wide Web*	–
XHTML	*Extensible HyperText Markup Language*	Linguagem de Marcação de Hipertexto Extensível
XML	*Extensible Markup Language*	Linguagem de Marcação Extensível
XSL	*Extensible Style Language*	Linguagem de Estilo Extensível

SUMÁRIO

Capítulo 1 Introdução 1

Capítulo 2 Camada de Aplicação 33

Capítulo 3 Camada de Transporte 139

Capítulo 6 Redes sem Fio e IP Móvel 471

Capítulo 7 Camada Física e Meios de Transmissão 539

Capítulo 8 Multimídia e Qualidade de Serviço 599

Capítulo 9 Gerenciamento de redes 693

Capítulo 10 Segurança de Redes 725

Capítulo 11 Programação de *Sockets* em Java 799

CAPÍTULO

1 INTRODUÇÃO

A maior rede de computadores do mundo, a internet, tem mais de um bilhão de usuários. Usando meios de transmissão com e sem fio, esse sistema conecta computadores de pequeno e grande porte, permite que usuários compartilhem uma enorme quantidade de informações, enviem mensagens uns aos outros, incluindo textos, imagens, áudio e vídeo. Explorar esse vasto sistema é o objetivo principal deste livro. Neste capítulo, entretanto, temos dois objetivos. O primeiro é fornecer uma visão geral da internet como uma *internetwork* (rede de redes) e discutir os componentes que a compõem. Parte do nosso primeiro objetivo é também introduzir o modelo de camadas de protocolos e adquirir uma visão inicial sobre a pilha de protocolos TCP/IP. Em outras palavras, o primeiro objetivo é preparar o leitor para os demais capítulos do livro. O nosso segundo objetivo é proporcionar informações pertinentes que não são, entretanto, necessárias para compreender o restante do livro. Dividimos este capítulo em quatro seções.

- Na primeira seção, introduzimos o conceito de redes locais (LANs – Local Area Networks) e redes de longa distância (WANs – Wide Area Networks) e apresentamos as definições básicas destes dois tipos de redes. Definimos uma *internetwork*, ou internet, como uma combinação de LANs e WANs. Também mostramos como uma organização pode criar uma internet privada por meio da conexão de suas LANs usando WANs. Finalmente, introduzimos a internet como um conjunto de redes globais composta por *backbones* (espinha dorsal), provedores de rede e redes de clientes.
- Na segunda seção, usamos o conceito de protocolos em camadas para mostrar como a tarefa a ser realizada pela internet é dividida em tarefas menores. Discutimos as cinco camadas que compõem a pilha de protocolos TCP/IP e introduzimos seus atributos e os protocolos nelas envolvidos. Discutimos também dois conceitos desse modelo: encapsulamento/desencapsulamento e multiplexação/demultiplexação.
- Na terceira seção, mostramos uma breve história da internet para os leitores interessados. Essa seção pode ser ignorada sem que haja perda de continuidade do texto.
- Na quarta seção, apresentamos a administração da internet e definimos os seus padrões e respectivos tempos de vida. Essa seção é apenas informativa e não é necessária para compreender o restante do livro.

1.1 VISÃO GERAL DA INTERNET

Embora o objetivo deste livro seja discutir a internet, um sistema que interliga bilhões de computadores no mundo, não devemos enxergá-la como uma rede única, mas como uma **inter-rede** (*internetwork*), uma combinação de redes. Portanto, começamos nossa jornada definindo uma rede. Em seguida, mostraremos como podemos conectar redes para criar pequenas inter-redes e, por fim, mostraremos a estrutura da internet e abriremos o caminho para estudá-la nos próximos dez capítulos.

1.1.1 Redes

Uma **rede** é a interligação de um conjunto de dispositivos capazes de se comunicar. Nesta definição, um dispositivo pode ser um ***host*** (ou um **sistema final**, como às vezes é chamado), tal como um grande computador, *desktop*, *laptop*, estação de trabalho, telefone celular ou sistema de segurança. Um dispositivo nessa definição também pode ser um **dispositivo de conexão**, tal como um roteador, que liga uma rede a outras redes, um *switch* (ou comutador) que liga dispositivos entre si, um modem (modulador-demodulador) que altera a forma dos dados, e assim por diante. Tais dispositivos em uma rede são conectados usando meios de transmissão com ou sem fio, como cabo ou ar. Quando conectamos dois computadores em casa usando um roteador *plug-and-play*, criamos uma rede, embora muito pequena.

Rede local

Uma **rede local** (LAN – Local Area Network) geralmente é uma propriedade privada e conecta alguns *hosts* em um único escritório, prédio ou *campus*. Dependendo das necessidades de uma organização, uma LAN pode ser simples com apenas dois computadores e uma impressora no escritório da casa de alguém, ou pode se estender por toda a empresa e incluir dispositivos de áudio e vídeo. Cada *host* em uma LAN possui um identificador, ou seja, um endereço que o define de forma unívoca na LAN. Um pacote enviado de um *host* para outro carrega tanto o endereço do *host* de origem como o de destino.

No passado, todos os *hosts* em uma rede eram conectados por meio de um cabo compartilhado, o que significava que um pacote enviado de um *host* para outro era recebido por todos os *hosts*. O destinatário correto armazenava o pacote; os outros o descartavam. Hoje, a maioria das LANs usa um mecanismo de conexão inteligente por meio de um *switch*, que é capaz de reconhecer o endereço de destino do pacote e encaminhá-lo a seu destino sem enviá-lo a todos os outros *hosts*. O *switch* alivia o tráfego na LAN e permite que mais de um par de *hosts* se comuniquem uns com os outros ao mesmo tempo, caso não haja uma fonte ou destino em comum entre cada par. Note que essa definição não define o número mínimo ou máximo de *hosts* em uma LAN. A Figura 1.1 mostra uma LAN usando um cabo compartilhado ou um *switch*.

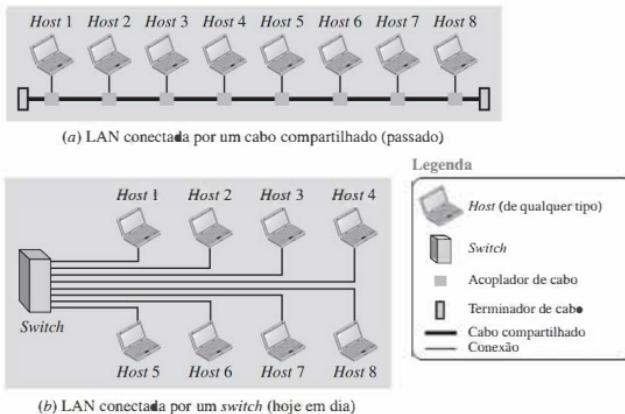



Figura 1.1 LAN isolada no passado e hoje em dia.

LANs serão discutidas em mais detalhes nos Capítulos 5 e 6.

Quando LANs eram usadas isoladamente (algo raro atualmente), eram projetadas para permitir que os recursos fossem compartilhados entre os *hosts*. Como veremos em breve, hoje em dia, LANs são conectadas umas às outras e com WANs (discutidas a seguir) para criar comunicações em um nível mais amplo.

Rede de longa distância

Uma **rede de longa distância** (WAN – Wide Area Network) é também uma interligação de dispositivos capazes de se comunicar. No entanto, existem algumas diferenças entre uma LAN e uma WAN. A LAN normalmente tem tamanho limitado, estendendo-se por um escritório, edifício ou *campus*, enquanto uma WAN tem uma extensão geográfica maior, abrangendo uma cidade, um estado, um pais, ou mesmo o mundo. Uma LAN interliga *hosts*; uma WAN interliga dispositivos de conexão como *switches*, roteadores ou modems. Uma LAN normalmente é propriedade privada da organização que a utiliza; uma WAN, por sua vez, costuma ser criada e operada por empresas de comunicação e alugada por uma organização que a utiliza. Encontramos dois exemplos distintos de WANs hoje em dia: WANs ponto a ponto e comutadas (ou WANs de comutação).

WAN ponto a ponto

Uma WAN ponto a ponto é uma rede que conecta dois dispositivos de comunicação usando um meio de transmissão (cabo ou ar). Veremos exemplos dessas WANs quando discutirmos as conexões entre as redes. A Figura 1.2 mostra um exemplo de uma WAN ponto a ponto.

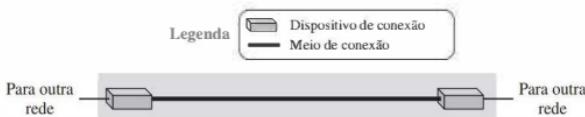



Figura 1.2 WAN ponto a ponto.

WAN comutada

Uma WAN comutada é uma rede com mais de duas extremidades. A WAN comutada, como veremos em breve, é utilizada no *backbone* das comunicações globais atuais. Pode-se dizer que uma WAN comutada é uma combinação de várias WANs ponto a ponto que são ligadas por meio de *switches*. A Figura 1.3 mostra um exemplo de uma WAN comutada.

Interconexão de redes: *internetwork*

Atualmente, é raro ver uma LAN ou WAN isolada, pois elas são conectadas umas às outras. Quando duas ou mais redes estão conectadas, elas criam uma ***internetwork*** ou **internet**. Como exemplo, suponha que uma organização tem dois escritórios: um na costa oeste e outro na costa leste dos Estados Unidos. Cada escritório tem uma LAN que permite que todos os funcionários naquele escritório se comuniquem entre si. Para que a comunicação entre funcionários de diferentes escritórios seja possível, os gestores decidem alugar uma WAN dedicada ponto a ponto de um provedor de serviço, como uma empresa de telefonia, e conectam as duas LANs. Agora, a empresa tem uma *internetwork*, ou uma internet privada (com *i* minúsculo). A comunicação entre os escritórios agora é possível. A Figural.4 mostra essa internet.

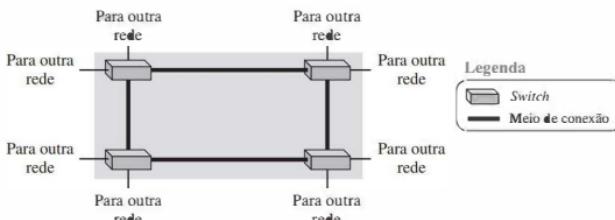



Figura 1.3 WAN comutada.

WANs serão discutidas em mais detalhes nos Capítulos 5 e 6.

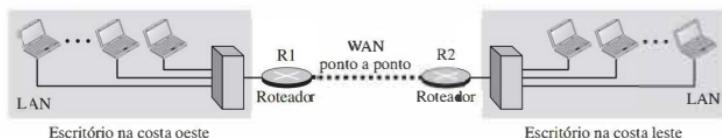



Figura 1.4 *Internetwork* composta de duas LANs e uma WAN ponto a ponto.

Quando um *host* no escritório da costa oeste envia uma mensagem para outro *host* no mesmo escritório, o roteador bloqueia a mensagem, mas o *switch* a direciona para seu destino. Por outro lado, quando um *host* na costa oeste envia uma mensagem a um *host* na costa leste, o roteador R1 encaminha o pacote para o roteador R2, e o pacote chega ao seu destino.

A Figura 1.5 mostra outra internet com várias LANs e WANs conectadas. Uma das WANs é uma WAN comutada com quatro *switches*.

1.1.2 Comutação

Dissemos que uma internet é uma combinação de enlaces (ou *links*) e *switches*, como os *switches* da camada de enlace e os roteadores que usamos nas seções anteriores. Na verdade, uma internet é uma **rede comutada** na qual um *switch* conecta pelo menos dois enlaces. Um *switch* precisa encaminhar dados vindos de um enlace para outro quando necessário. Os dois tipos mais comuns de redes de comutação são as redes de comutação de circuitos e as redes de comutação de pacotes, que serão discutidas a seguir.

Rede de comutação de circuitos

Em uma **rede de comutação de circuitos**, uma conexão dedicada, chamada de circuito, está sempre disponível entre os dois sistemas finais; o *switch* apenas ativa ou desativa esta conexão. A Figura 1.6 mostra uma rede comutada muito simples que conecta quatro telefones em cada uma de suas extremidades. Usamos aparelhos telefônicos em vez de computadores como sistemas finais porque,

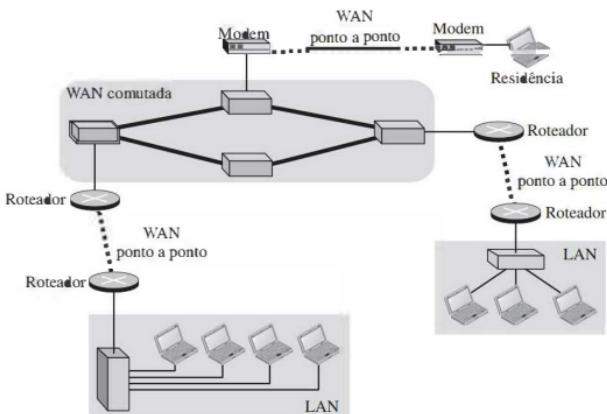



Figura 1.5 Rede heterogênea composta por quatro WANs e três LANs.

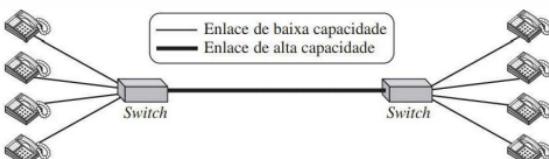



Figura 1.6 Rede de comutação de circuitos.

no passado, a comutação de circuitos era muito comum nas redes telefônicas, embora parte da rede de telefonia atual seja uma rede de comutação de pacotes.

Na figura anterior, os quatro telefones de cada lado são ligados a um *switch* que liga um aparelho telefônico de um lado a um aparelho telefônico no outro lado. A linha grossa ligando os dois *switches* é um enlace de comunicação de alta capacidade que pode manter quatro transmissões de voz ao mesmo tempo; essa capacidade pode ser compartilhada por todos os pares de aparelhos telefônicos. Os *switches* usados no exemplo podem executar tarefas de encaminhamento, mas não têm capacidade de armazenamento.

Vejamos dois casos. No primeiro, todos os aparelhos telefônicos estão ocupados; quatro pessoas de um lado estão falando com quatro pessoas do outro; a capacidade da linha grossa é totalmente utilizada. No segundo caso, apenas um aparelho telefônico de um lado está ligado a um aparelho telefônico do outro; apenas um quarto da capacidade da linha grossa é usado. Isso significa que uma rede de comutação de circuitos é eficiente apenas quando está trabalhando com sua capacidade total; na maior parte do tempo, ela é ineficiente porque trabalha apenas com capacidade parcial. A razão pela qual é necessário fazer com que a capacidade da linha grossa seja quatro vezes a capacidade de cada enlace de voz é que não queremos que a comunicação

falhe quando todos os aparelhos telefônicos de um lado forem conectados com todos os aparelhos telefônicos do outro.

Rede de comutação de pacotes

Em uma rede de computadores, a comunicação entre dois sistemas finais é feita usando blocos de dados chamados **pacotes**. Em outras palavras, em vez da comunicação contínua que observamos entre dois aparelhos telefônicos quando estão sendo usados, vemos a troca de pacotes de dados individuais entre os dois computadores. Isso permite que os *switches* desempenhem funções tanto de armazenamento como de encaminhamento, porque cada pacote é uma entidade independente que pode ser armazenada e enviada posteriormente. A Figura 1.7 mostra uma pequena rede de comutação de pacotes que conecta quatro computadores de um local com quatro computadores de outro local.

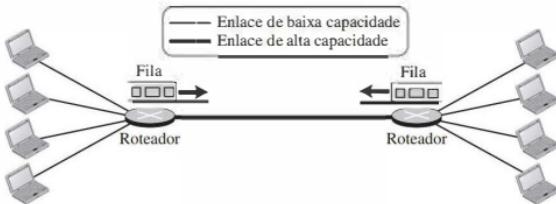



Figura 1.7 Rede de comutação de circuitos.

Um roteador em uma rede de comutação de pacotes tem uma fila que pode armazenar e encaminhar pacotes. Agora, suponha que a capacidade da linha grossa seja apenas duas vezes a capacidade da linha de dados conectando os computadores aos roteadores. Se apenas dois computadores (um de cada lado da rede) precisam se comunicar um com o outro, não há espera na entrega dos pacotes. No entanto, se os pacotes chegam a um roteador quando a linha grossa já está com sua capacidade total ocupada, os pacotes devem ser armazenados e encaminhados na ordem em que chegarem. Esses dois exemplos simples mostram que uma rede de comutação de pacotes é mais eficiente do que uma rede de comutação de circuitos, mas que os pacotes podem sofrer alguns atrasos.

Neste livro, discutimos principalmente sobre as redes de comutação de pacotes. No Capítulo 4, falaremos sobre redes de comutação de pacotes com mais detalhes, discutindo o desempenho de tais redes.

1.1.3 A Internet

Conforme discutimos anteriormente, uma internet (note o *i* minúsculo) consiste em duas ou mais redes que podem comunicar-se entre si. A internet mais conhecida é aquela chamada de *Internet* (com *I maiúsculo*), composta por milhares de redes interconectadas. A Figura 1.8 mostra uma visão conceitual (não geográfica) da Internet.

A figura mostra a Internet como vários *backbones*, redes de provedores e redes de clientes. No nível superior, os *backbones* (as espinhas dorsais da Internet) são redes grandes e de propriedade de algumas empresas de comunicação, como Sprint, Verizon (MCI), AT&T, e NTT. As redes de *backbone* são conectadas através de alguns sistemas complexos de comutação, chamados *pontos de troca de tráfego*. No segundo nível, encontram-se redes menores chamadas *redes de provedores*,

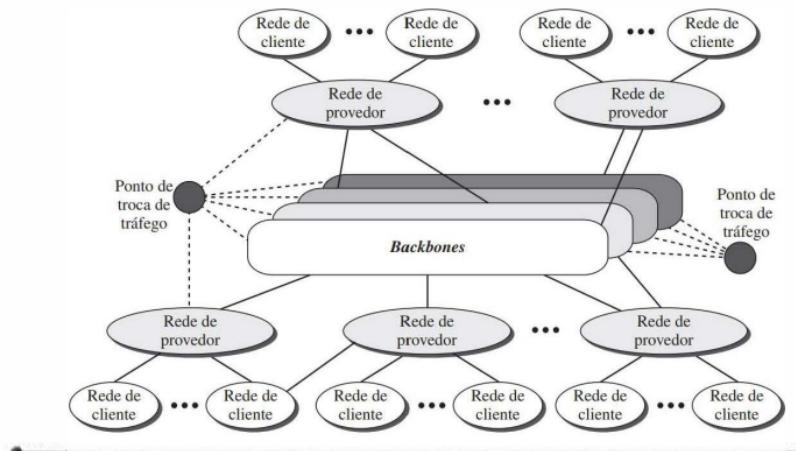



Figura 1.8 Internet hoje em dia.

que usam os serviços dos *backbones* mediante o pagamento de uma taxa. As redes de provedores estão conectadas aos *backbones* e algumas vezes a outras redes de provedores. As *redes de clientes* são as redes na borda da Internet que efetivamente usam os serviços prestados pela Internet. Elas pagam taxas para as redes de provedores para receber serviços.

Os *backbones* e as redes de provedores são também chamados **Provedores de Serviços de Internet** (ISPs – Internet Service Providers). Os *backbones* são frequentemente denominados ISPs internacionais e as redes de provedores, ISPs nacionais ou regionais.

1.1.4 Acessando a Internet

A Internet atual é um conjunto de redes que permite que qualquer usuário se torne parte dela, desde que esteja fisicamente conectado a um ISP. A ligação física costuma ser feita por meio de uma WAN ponto a ponto. Nesta seção, descrevemos brevemente como isso pode acontecer, mas adiamos a apresentação dos detalhes técnicos dessa conexão para os Capítulos 6 e 7.

Usando redes de telefonia

Atualmente, a maioria das residências e pequenas empresas têm serviços de telefonia, o que significa que estão conectadas a uma rede telefônica. Uma vez que a maioria das redes de telefonia já está conectada à Internet, uma opção de conexão para residências e pequenas empresas é trocar a linha de voz e a central telefônica por uma WAN ponto a ponto. Isso pode ser feito de duas maneiras.

- **Serviço discado.** Também conhecido como *dial-up*, a primeira solução é adicionar um modem à linha telefônica para converter dados em voz. O *software* instalado no computador faz uma ligação para o ISP imitando uma ligação telefônica. Infelizmente, o serviço discado é muito lento, e enquanto a linha é usada para conexão com a Internet, ela não pode ser usada para ligações telefônicas (voz). Esse recurso só é útil para pequenas

residências e empresas com ligação ocasional com a Internet. Discutiremos o serviço discado no Capítulo 5.

- **Serviço DSL.** Desde o advento da Internet, algumas companhias telefônicas têm atualizado suas linhas para prover serviços de Internet com maior velocidade para residências ou pequenas empresas. O serviço DSL também permite que a linha seja usada simultaneamente para comunicação de voz e dados. Discutiremos o serviço DSL no Capítulo 5.

Usando redes a cabo

Nas últimas duas décadas, cada vez mais consumidores começaram a usar os serviços de TV a cabo em vez de antenas para receber transmissões de TV. As empresas de TV a cabo têm atualizado as suas redes de cabo e criado conexões à Internet. Uma residência ou uma pequena empresa pode ser conectada à Internet usando este serviço que fornece uma velocidade de conexão mais elevada, mas a velocidade pode sofrer variações de acordo com o número de vizinhos que usam o mesmo cabo. Discutiremos as redes de cabo no Capítulo 5.

Usando redes sem fio

Recentemente, a conectividade sem fios tem se tornado cada vez mais popular. Uma casa ou uma pequena empresa pode usar uma combinação de conexões sem fio e cabeadas para acessar a Internet. Com o acesso crescente a WANs sem fio, através delas uma casa ou uma pequena empresa pode ser conectada à Internet. Discutiremos acesso sem fio no Capítulo 6.

Conexão direta à Internet

Uma grande organização ou uma grande corporação pode se tornar sozinha um ISP local e se conectar à Internet. Isso pode ser feito se a empresa alugar uma WAN de alta velocidade de um provedor de serviços de telecomunicações e conectar-se a um ISP regional. Por exemplo, uma grande universidade com vários *campi* pode criar uma *internetwork* e então conectar esta *internetwork* à Internet.

1.1.5 *Hardware* e *Software*

Apresentamos uma visão geral da estrutura da Internet, composta de redes grandes e pequenas, conectadas entre si por meio de dispositivos de conexão. Deve ficar claro, entretanto, que nada vai acontecer se simplesmente conectarmos essas peças. Para que a comunicação aconteça, precisamos tanto de *hardware* como de *software*. Isso é semelhante a uma computação complexa, na qual é necessário tanto um computador como um programa. Na próxima seção, mostramos como estas combinações de *hardware* e *software* são coordenadas entre si usando *protocolos em camadas*.

1.2 PROTOCOLO EM CAMADAS

Uma palavra que ouvimos o tempo todo quando falamos sobre a Internet é *protocolo*. Um **protocolo** define as regras que o remetente, o destinatário e todos os dispositivos intermediários precisam seguir para que sejam efetivamente capazes de se comunicar. Quando a comunicação é simples, talvez seja necessário apenas um protocolo simples; quando a comunicação é complexa, talvez seja preciso dividir a tarefa em diferentes camadas, tornando necessário o uso de um protocolo em cada camada, ou um **protocolo em camadas**.

1.2.1 Cenários

Vamos descrever dois cenários simples para entender melhor a necessidade de um protocolo em camadas.

Primeiro cenário

No primeiro cenário, a comunicação é tão simples que pode acontecer em apenas uma camada. Suponha que Maria e Ana são vizinhas com diversas ideias em comum. A comunicação entre elas ocorre em uma camada, face a face, na mesma língua, como mostrado na Figura 1.9.

Figura 1.9 Protocolo em uma camada.

Até mesmo neste cenário simples, podemos ver que um conjunto de regras deve ser seguido. Primeiro, Maria e Ana sabem que devem cumprimentar uma à outra quando se encontram. Segundo, sabem que devem limitar seu vocabulário ao nível de sua amizade. Terceiro, cada uma delas sabe que deve se calar enquanto a outra estiver falando. Quarto, cada uma delas sabe que a conversa deve ser um diálogo, não um monólogo: ambas devem ter oportunidade de falar sobre o assunto em questão. Quinto, elas devem trocar algumas palavras agradáveis quando partirem.

Podemos ver que o protocolo usado por Maria e Ana é diferente da comunicação entre um professor e os alunos em uma sala de aula. A comunicação no segundo caso é mais um monólogo; o professor fala a maior parte do tempo, exceto quando um aluno tem uma pergunta, situação na qual o protocolo dita que o aluno deve levantar a mão e aguardar permissão para falar. Neste caso, a comunicação costuma ser muito formal e limitada ao assunto ensinado.

Segundo cenário

No segundo cenário, vamos supor que Ana recebeu uma oferta para assumir um cargo mais elevado na sua empresa, mas precisa se deslocar para outra filial localizada em uma cidade muito distante de Maria. As duas amigas ainda querem continuar a se comunicar e trocar ideias porque conceberam um projeto inovador para iniciar um novo negócio quando ambas se aposentarem. Elas decidem continuar suas conversas usando cartas enviadas por correio. No entanto, não querem que suas ideias sejam reveladas a outras pessoas caso as cartas sejam interceptadas. Elas entram em acordo sobre uma técnica de cifração/decifração. A remetente da carta cifra seu conteúdo para torná-lo ilegível para um intruso; a destinatária da carta decifra o conteúdo para obter o texto original. Discutiremos métodos de cifração/decifração no Capítulo 10, mas por ora, digamos que Maria e Ana usam uma técnica que torna difícil decifrar a carta sem os códigos. Agora podemos dizer que a comunicação entre Maria e Ana envolve três camadas, conforme mostrado na Figura 1.10. Supomos que Ana e Maria têm, cada uma, três máquinas (ou robôs) que podem executar a tarefa de cada camada.

Vamos supor que Maria envia a primeira carta para Ana. Maria fala com a máquina na terceira camada, como se a máquina fosse Ana e a estivesse ouvindo. A máquina de terceira camada escuta o que Maria diz e cria o texto original (uma carta em português, às claras), que é passada para a máquina da segunda camada. A máquina da segunda camada pega o texto

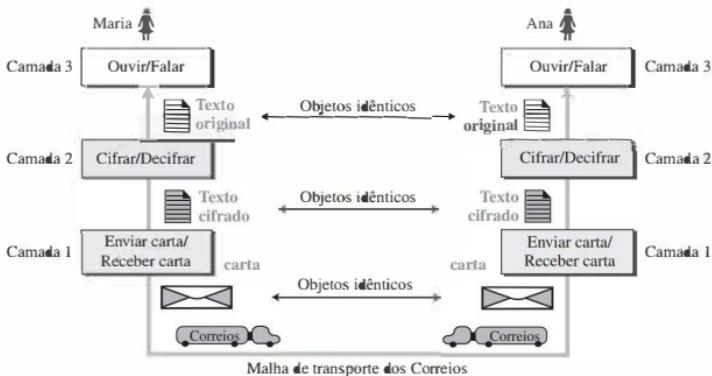



Figura 1.10 Protocolo de três camadas.

original, aplica o método de cifração, criando o texto cifrado que é passado para a máquina da primeira camada. A máquina da primeira camada, presumivelmente um robô, pega o texto cifrado, coloca-o em um envelope, adiciona os endereços do remetente e do destinatário, e envia o envelope pelo correio.

No lado de Ana, a máquina da primeira camada pega a carta da caixa de correio dela, reconhecendo a carta de Maria pelo endereço do remetente. A máquina retira o texto cifrado do envelope e o entrega à máquina da segunda camada. A máquina da segunda camada decifra a mensagem, criando o texto original, e passa este para a máquina da terceira camada. A máquina da terceira camada pega o texto original e o lê como se fosse Maria quem estivesse falando.

O uso de protocolos em camadas nos permite dividir uma tarefa complexa em várias tarefas menores e mais simples. Por exemplo, na Figura 1.10, poderíamos ter usado apenas uma máquina para fazer o trabalho de todas as três máquinas. No entanto, se Maria e Ana decidirem que a cifração/decifração feita por esta máquina não é suficiente para proteger seu segredo, elas teriam que trocar a máquina inteira. Já da forma apresentada, elas precisariam trocar apenas a máquina da segunda camada; as outras duas podem continuar sendo as mesmas. Isto é chamado de *modularidade*. A modularidade neste caso significa que as camadas são independentes. Uma camada (módulo) pode ser definida como uma caixa preta com entradas e saídas, sem qualquer preocupação sobre como as entradas são transformadas em saídas. Se duas máquinas fornecem as mesmas saídas para as mesmas entradas, elas podem substituir uma à outra. Por exemplo, Ana e Maria podem comprar uma máquina da segunda camada de dois fabricantes diferentes. Contanto que as duas máquinas criem o mesmo texto cifrado a partir do mesmo texto original e vice-versa, elas cumprem o seu trabalho.

Uma das vantagens da divisão de protocolos em camadas é que isto nos permite separar os serviços de suas implementações. Uma camada deve ser capaz de receber um conjunto de serviços da camada inferior e de prover serviços para a camada superior; não importando como a camada é implementada. Por exemplo, Maria pode decidir não comprar a máquina (robô) para a primeira camada; ela pode fazer o trabalho sozinha. Contanto que Maria execute as tarefas previstas pela primeira camada, em ambos os sentidos, o sistema de comunicação funcionará.

Outra vantagem de usar um protocolo em camadas, a qual não pode ser vista em nossos exemplos simples, mas aparece quando discutimos protocolo em camadas na Internet, é que a comunicação nem sempre usa apenas dois sistemas finais; existem sistemas intermediários que precisam apenas de algumas camadas, não de todas. Se não fossem usados os protocolos em camadas,

teríamos que tornar cada sistema intermediário tão complexo quanto os sistemas finais, o que deixaria todo o sistema mais caro.

Existe alguma desvantagem em usar protocolos em camadas? Pode-se argumentar que ter uma única camada torna o trabalho mais fácil. Não há necessidade de fazer com que cada camada forneça um serviço para a camada superior e passe serviço para a camada inferior. Por exemplo, Ana e Maria poderiam encontrar ou construir uma máquina que pudesse realizar todas as três tarefas. No entanto, como mencionado anteriormente, se um dia elas descobrissem que o seu código foi quebrado, cada uma teria que substituir toda a máquina com um novo código em vez de mudar apenas a máquina na segunda camada.

Princípios de protocolos em camadas

Vamos discutir alguns princípios de protocolos em camadas. O primeiro princípio dita que, se quisermos que a comunicação seja bidirecional, precisamos projetar cada camada de modo que ela seja capaz de executar duas tarefas opostas, uma em cada direção. Por exemplo, a tarefa da terceira camada é *ouvir* (em um sentido) e *falar* (no outro sentido). A segunda camada deve ser capaz de cifrar e decifrar. A primeira camada precisa enviar e receber cartas.

O segundo princípio importante é que os dois objetos em cada camada em ambos os lados devem ser idênticos. Por exemplo, o objeto na camada 3 em ambos os lados deve ser uma carta em texto original; na camada 2, em ambos os lados o objeto deve ser uma carta contendo texto cifrado, e na camada 1, em ambos os lados o objeto deve ser uma carta.

Conexões lógicas

Após seguir os dois princípios anteriores, podemos pensar sobre a conexão lógica entre cada camada, como mostrado na Figura 1.11. Isso significa que temos comunicação de uma camada para a outra. Maria e Ana podem pensar como se existisse uma conexão (imaginária) em cada camada lógica, através da qual elas podem enviar o objeto criado daquela camada. Veremos que o conceito de conexão lógica vai nos ajudar a entender melhor a tarefa das camadas que encontramos nas comunicações e redes de dados.

Figura 1.11 Conexão lógica entre camadas em sistemas finais.

1.2.2 A pilha de protocolos TCP/IP

Agora que conhecemos o conceito de protocolos em camadas e discutimos a comunicação lógica entre as camadas do nosso segundo cenário, podemos introduzir o TCP/IP (Transmission Control Protocol/Internet Protocol, ou Protocolo de Controle de Transmissão/Protocolo Internet).

O TCP/IP consiste em uma pilha de protocolos (um conjunto de protocolos organizados em diferentes camadas) usados na Internet atual. É um protocolo hierárquico composto de módulos interativos, cada um dos quais provendo uma funcionalidade específica. O termo *hierárquico* significa que cada protocolo do nível superior é apoiado pelos serviços fornecidos por um ou mais protocolos dos níveis abaixo dele. A pilha de protocolos TCP/IP original foi definida como quatro camadas de *software* construídas sobre o *hardware*. Hoje, no entanto, o TCP/IP é visto como um modelo de cinco camadas. A Figura 1.12 mostra ambas as configurações.

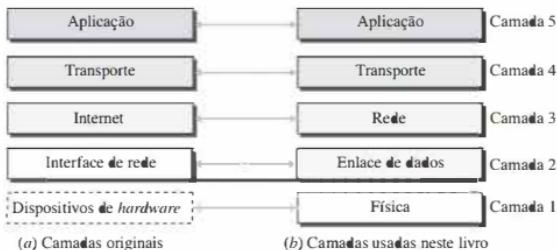



Figura 1.12 Camadas na pilha de protocolos TCP/IP.

Arquitetura em camadas

Para mostrar como as camadas da pilha de protocolos TCP/IP estão envolvidas na comunicação entre dois *hosts*, assumimos que queremos usar esta pilha em uma pequena internet composta de três LANs, cada uma contendo um *switch* de camada de enlace. Também assumimos que os enlaces estão conectados por um roteador, como mostrado na Figura 1.13.

Vamos supor que o computador A se comunica com o computador B. Como mostra a figura, temos cinco dispositivos comunicando-se na presente comunicação: *host* de origem (computador A), o *switch* de camada de enlace no enlace 1, o roteador, o *switch* de camada de enlace no enlace 2, e o *host* de destino (computador B). Cada dispositivo está envolvido com um conjunto de camadas, dependendo do papel daquele dispositivo na Internet. Os dois *hosts* estão envolvidos em todas as cinco camadas; o *host* de origem precisa criar uma mensagem na camada de aplicação e enviá-la para as camadas inferiores de modo que esta mensagem seja fisicamente enviada ao *host* de destino. O *host* de destino precisa receber a comunicação na camada física e então entregá-la através das outras camadas para a camada de aplicação.

O roteador está envolvido em apenas três camadas; não há uma camada de transporte ou de aplicação em um roteador quando este é usado apenas para roteamento. Apesar de os roteadores estarem sempre envolvidos em uma camada de rede, há *n* combinações de camadas de enlace e físicas, onde *n* é o número de enlaces aos quais o roteador está conectado. A razão é que cada enlace pode usar o seu próprio protocolo da camada de enlace de dados ou física. Por exemplo, na figura anterior, o roteador está envolvido em três enlaces, mas a mensagem enviada da origem A para o destino B está envolvida em dois. Cada enlace pode estar usando diferentes protocolos da camada de enlace e da camada física; o roteador precisa receber um pacote do enlace 1 usando um par de protocolos e entregá-lo ao enlace 2 usando outro par de protocolos.

Um *switch* de camada de enlace em um enlace, por outro lado, está envolvido apenas em duas camadas, a de enlace de dados e a física. Apesar de cada *switch* na figura anterior possuir duas conexões diferentes, as conexões estão no mesmo enlace, o qual usa apenas um conjunto de protocolos. Isso significa que, ao contrário de um roteador, um *switch* de camada de enlace está envolvido apenas em um enlace de dados e em uma camada física.

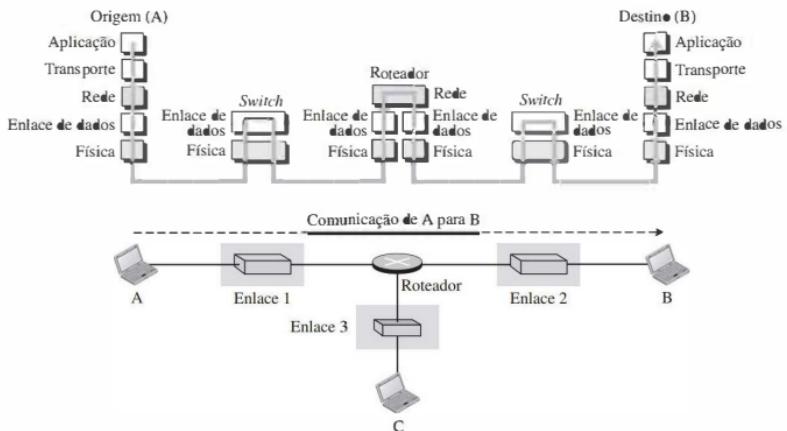



Figura 1.13 Comunicação através de uma internet.

Camada na pilha de protocolos TCP/IP

Depois da introdução anterior, vamos discutir brevemente as funções e deveres das camadas da pilha de protocolos TCP/IP. Cada camada é discutida em detalhes nos próximos seis capítulos do livro. Para entender melhor as funções de cada camada, precisamos refletir sobre as conexões lógicas entre as camadas. A Figura 1.14 mostra as conexões lógicas na nossa internet simples.

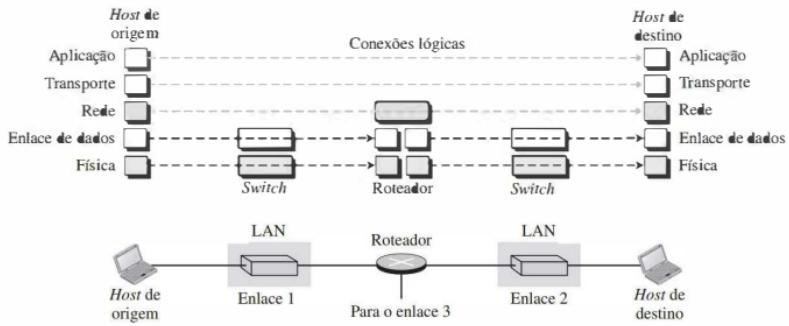



Figura 1.14 Conexões lógicas entre camadas da pilha de protocolos TCP/IP.

O uso de conexões lógicas facilita a reflexão sobre a função de cada camada. Como mostra a figura, a função das camadas de aplicação, transporte e de rede é fim a fim. No entanto, a função das camadas de enlace de dados e física é salto a salto, sendo que um salto (ou *hop*) é um *host* ou

roteador. Em outras palavras, o domínio da ação das três camadas mais altas é a internet e o domínio de ação das duas camadas mais baixas é o enlace.

Outra maneira de enxergar as conexões lógicas é considerando a unidade de dados criada a partir de cada camada. Nas três camadas superiores, a unidade de dados (pacotes) não deve ser alterada por qualquer roteador ou *switch* de camada de enlace. Nas duas camadas inferiores, o pacote criado pelo *host* é alterado apenas pelos roteadores, e não pelos *switches* de camada de enlace.

A Figura 1.15 mostra o segundo princípio discutido anteriormente para protocolos em camadas. Mostramos os objetos idênticos abaixo de cada camada relacionada com cada dispositivo.

Note que, embora a conexão lógica na camada de rede seja entre os dois *hosts*, só podemos dizer que objetos idênticos existem entre dois saltos nesse caso, porque um roteador pode fragmentar o pacote na camada de rede e enviar mais pacotes do que recebeu (ver fragmentação no Capítulo 4). Note que o enlace entre dois saltos não altera o objeto.

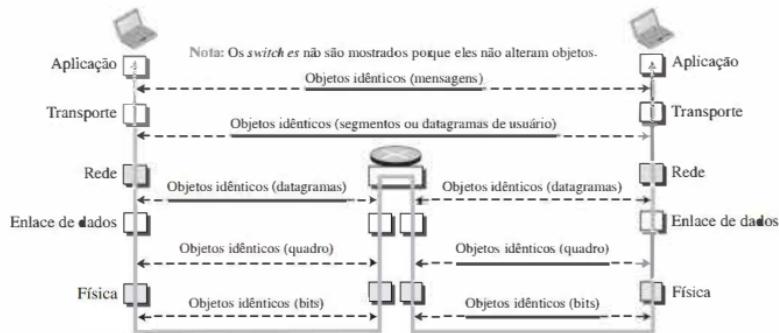



Figura 1.15 Objetos idênticos na pilha de protocolos TCP/IP.

Descrição de cada camada no TCP/IP

Entendido o conceito de comunicação lógica, estamos prontos para discutir brevemente a função de cada camada. Nossa discussão neste capítulo será bastante breve, mas voltaremos à função de cada camada nos próximos seis capítulos.

Camada de aplicação

Como mostrado na Figura 1.14, a conexão lógica entre as duas camadas de aplicação é fim a fim. As duas camadas de aplicação trocam ***mensagens*** entre si como se não houvesse uma ponte entre elas. Entretanto, é importante ter em mente que a comunicação é feita através de todas as camadas.

A comunicação na camada de aplicação se dá entre dois *processos* (dois programas em execução nessa camada). Para se comunicar, um processo envia um pedido para o outro processo e recebe uma resposta. A comunicação processo a processo é a função da camada de aplicação. A camada de aplicação na Internet inclui muitos protocolos predefinidos, mas o usuário também pode criar um par de processos para serem executados nos dois *hosts*. No Capítulo 2, vamos explorar essa situação.

O Protocolo de Transferência de Hipertexto (HTTP – HyperText Transfer Protocol) é um meio de acesso à World Wide Web (WWW). O Protocolo Simples de Transferência de Correio (SMTP – Simple Mail Transfer Protocol) é o principal protocolo utilizado no serviço de correio

eletrônico (e-mail). O Protocolo de Transferência de Arquivos (FTP – File Transfer Protocol) é usado para transferir arquivos de um *host* para outro. A Rede de Terminais (TELNET – Terminal Network) e o SSH (Secure Shell) são usados para acessar uma máquina remotamente. O Protocolo Simples de Gerenciamento de Rede (SNMP – Simple Network Management Protocol) é usado por um administrador para gerenciar a Internet nos níveis global e local. O Sistema de Nomes de Domínio (DNS – Domain Name System) é usado por outros protocolos para localizar o endereço de camada de rede de um computador. O Protocolo de Gerenciamento de Grupos Internet (IGMP – Internet Group Management Protocol) é usado para agregar participantes a um grupo. Discutiremos a maioria dos protocolos no Capítulo 2 e alguns em outros capítulos.

Camada de transporte

A conexão lógica na camada de transporte também é fim a fim. A camada de transporte no *host* de origem recebe a mensagem da camada de aplicação, a encapsula em um pacote da camada de transporte (denominado um *segmento* ou um *datagrama de usuário*, dependendo do protocolo) e a envia através da conexão lógica (imaginária) para a camada de transporte no *host* de destino. Em outras palavras, a camada de transporte é responsável por prover serviços para a camada de aplicação: para obter uma mensagem de um aplicativo sendo executado no *host* de origem e entregá-la para o aplicativo correspondente no *host* de destino. Podemos perguntar: por que precisamos de uma camada de transporte fim a fim quando já temos uma camada de aplicação fim a fim? O motivo é a separação de tarefas e deveres, algo que discutimos anteriormente. A camada de transporte deve ser independente da camada de aplicação. Além disso, vamos ver que temos mais de um protocolo na camada de transporte, o que significa que cada aplicativo pode usar o protocolo que mais satisfaça suas necessidades.

Como dissemos, existem alguns protocolos da camada de transporte na Internet, cada um projetado para alguma tarefa específica. O principal protocolo, o TCP (Transmission Control Protocol, ou Protocolo de Controle de Transmissão), é um protocolo orientado à conexão que inicialmente estabelece uma conexão lógica entre as camadas de transporte dos dois *hosts* antes de transferir dados. Ele cria um canal lógico entre duas camadas TCP para transferir um fluxo de *bytes*. O TCP provê controle de fluxo (harmonizando a taxa de envio de dados do *host* de origem e a taxa de recepção de dados do *host* de destino para impedir que este fique sobrecarregado), controle de erros (para garantir que os segmentos cheguem ao destino sem erros e para reenviar segmentos corrompidos) e controle de congestionamento para reduzir a perda de segmentos devido a congestionamentos na rede. O outro protocolo comum, o UDP (User Datagram Protocol, ou Protocolo de Datagramas de Usuário), é um protocolo não orientado à conexão que transmite datagramas de usuário sem antes criar uma conexão lógica. No UDP, cada datagrama de usuário é uma entidade independente sem qualquer relação com os datagramas de usuário anteriores ou próximos (o significado do termo *não orientado à conexão*). O UDP é um protocolo simples que não fornece controle de fluxo, erros, ou congestionamento. A sua simplicidade, que se traduz em reduzido *overhead* (carga computacional adicional), é atraente para um aplicativo que precisa enviar mensagens curtas e não consegue realizar a retransmissão de pacotes envolvida no TCP quando um pacote é corrompido ou perdido. Um protocolo mais recente, o SCTP (Stream Control Transmission Protocol, ou Protocolo de Controle de Fluxo de Transmissão) foi concebido para atender as novas aplicações que estão surgindo na área de multimídia. Vamos discutir UDP e TCP no Capítulo 3 e SCTP no Capítulo 8.

Camada de rede

A camada de rede é responsável por criar uma conexão entre o computador de origem e o computador de destino. A comunicação na camada de rede é *host* a *host*. No entanto, uma vez que pode haver vários roteadores da origem ao destino, os roteadores no caminho são responsáveis por escolher a melhor rota para cada pacote. Podemos dizer que a camada de rede é responsável pela comunicação *host* a *host* e pelo roteamento de pacotes através de possíveis rotas. Novamente, podemos nos perguntar por que precisamos da camada de rede. Poderíamos ter adicionado a função de roteamento à camada de transporte e removido esta camada. Um dos motivos, como dissemos

antes, é a separação das diferentes tarefas entre as diferentes camadas. A segunda razão é que assim os roteadores não precisam das camadas de aplicação e de transporte. Separar as tarefas nos permite usar um menor número de protocolos nos roteadores.

A camada de rede da Internet inclui o protocolo principal, chamado IP (Internet Protocol, ou Protocolo Internet), que define o formato do pacote, denominado datagrama na camada de rede. O IP também define o formato e a estrutura de endereços usados nesta camada. O IP também é responsável pelo roteamento de um pacote de sua origem até seu destino, o que é conseguido por meio do encaminhamento do datagrama de cada roteador para o próximo roteador no seu caminho.

O IP é um protocolo não orientado à conexão que não fornece qualquer serviço de controle de fluxo, controle de erro, ou controle de congestionamento. Isto significa que se qualquer um destes serviços for necessário para um aplicativo, este deve usar os serviços do protocolo da camada de transporte. A camada de rede também inclui protocolos de roteamento *unicast* (um para um) e *multicast* (um para muitos). Um protocolo de roteamento não participa no roteamento (que é de responsabilidade do IP), mas cria tabelas de roteamento para os roteadores de modo a ajudá-los no processo de roteamento.

A camada de rede também apresenta alguns protocolos auxiliares que ajudam o IP nas tarefas de entrega e roteamento. O ICMP (Internet Control Message Protocol, ou Protocolo de Mensagens de Controle da Internet) ajuda o IP a relatar alguns problemas durante o roteamento de pacotes. O IGMP (Internet Group Management Protocol, ou Protocolo de Gerenciamento de Grupos Internet) é outro protocolo que ajuda o IP, neste caso em tarefas de *multicast*. O DHCP (Dynamic *Host* Configuration Protocol, ou Protocolo de Configuração Dinâmica de *Host*) ajuda o IP a obter o endereço de camada de rede para um *host*. O ARP (Address Resolution Protocol, ou Protocolo de Resolução de Endereços) é um protocolo que ajuda o IP a localizar o endereço da camada de enlace de um *host* ou de roteador quando o seu endereço de camada de rede é dado. Discutimos ICMP, IGMP e DHCP no Capítulo 4, mas discutiremos ARP no Capítulo 5.

Camada de enlace de dados

Vimos que uma internet é composta de vários enlaces (LANs e WANs) conectados por roteadores. Pode haver vários conjuntos sobrepostos de enlaces pelos quais um datagrama pode viajar da origem até o destino. Os roteadores são responsáveis por escolher os *melhores* enlaces, no entanto, quando o próximo enlace a ser utilizado é determinado pelo roteador, a camada de enlace de dados é responsável por pegar o datagrama e movê-lo através do enlace. Este pode ser uma LAN cabeada usando um *switch* de camada de enlace, uma LAN sem fio, uma WAN cabeada ou uma WAN sem fio. Pode também haver diferentes protocolos usados com qualquer tipo de enlace. Em cada caso, a camada de enlace de dados é responsável por mover o pacote através do enlace.

O TCP/IP não define um protocolo específico para a camada de enlace de dados; ele suporta todos os protocolos padronizados e proprietários. Qualquer protocolo que seja capaz de receber o datagrama e transportá-lo através do enlace já satisfaz as necessidades da camada de rede. A camada de enlace de dados pega um datagrama e o encapsula em um pacote chamado de quadro (também denominado *frame*).

Cada protocolo da camada de enlace pode fornecer um serviço distinto. Alguns protocolos da camada de enlace fornecem completa detecção e correção de erros, enquanto outros provêm apenas correção de erros. Discutiremos enlaces cabeados no Capítulo 5 e enlaces sem fios no Capítulo 6.

Camada física

Podemos dizer que a camada física é responsável por transportar os *bits* individuais de um quadro através do enlace. Embora a camada física seja o nível mais baixo na pilha de protocolos TCP/IP, a comunicação entre dois dispositivos na camada física é ainda uma comunicação lógica, pois há outra camada escondida sob a camada física, o meio de transmissão. Dois dispositivos são conectados por um meio de transmissão (cabo ou ar). Precisamos ter em mente que o meio de transmissão não carrega *bits*; ele carrega sinais elétricos ou ópticos. Assim, os *bits* recebidos em um quadro da camada de enlace de dados são transformados e enviados através

dos meios de transmissão, mas podemos pensar como se a unidade lógica entre duas camadas físicas em dois dispositivos fosse um *bit*. Existem diversos protocolos que transformam um *bit* em um sinal. Vamos discuti-los no Capítulo 7, quando discutiremos a camada física e os meios de transmissão.

Encapsulamento e desencapsulamento

Um conceito importante relacionado a protocolos em camadas na Internet é o encapsulamento/desencapsulamento. A Figura 1.16 mostra esse conceito para a pequena internet da Figura 1.13.

Omitimos as camadas para os *switches* na camada de enlace porque não ocorre qualquer encapsulamento/desencapsulamento nesses dispositivos. Na Figura 1.16, mostramos o encapsulamento no *host* de origem, desencapsulamento no *host* de destino e encapsulamento/desencapsulamento no roteador.

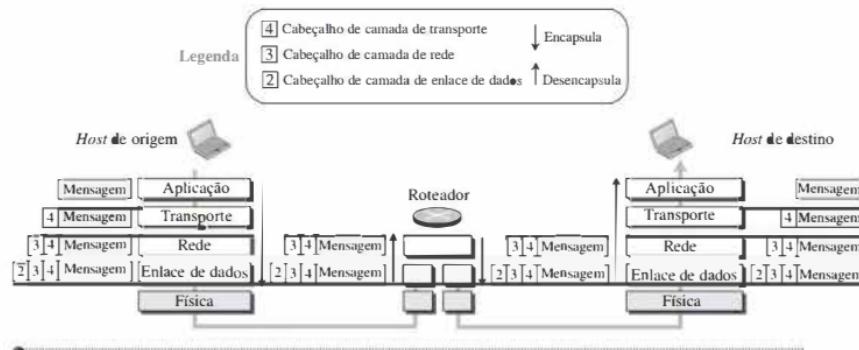



Figura 1.16 Encapsulamento/Desencapsulamento.

Encapsulamento no *host* de origem

Na origem, temos apenas o encapsulamento.

1. Na camada de aplicação, os dados a serem trocados são denominados uma *mensagem*. A mensagem normalmente não contém um *header* (cabeçalho) ou *trailer* (rodapé) mas, se for o caso, o conjunto completo de dados corresponde à mensagem. A mensagem é passada para a camada de transporte.
2. A camada de transporte trata a mensagem como o *payload*, a carga útil com a qual a camada de transporte deve lidar. Ela adiciona a este *payload* o cabeçalho da camada de transporte, o qual contém os identificadores do aplicativo de origem e destino que desejam se comunicar, além das informações extras necessárias para a entrega fim a fim da mensagem, tal como informações necessárias para controle de fluxo, erro ou congestionamento. O resultado é o pacote da camada de transporte, que é chamado de *segmento* (no TCP) ou de *datagrama de usuário* (no UDP). A camada de transporte passa, então, o pacote para a camada de rede.
3. A camada de rede trata o pacote da camada de transporte como *payload* e adiciona seu próprio cabeçalho a esta carga útil. O cabeçalho contém os endereços dos *hosts* de origem e destino, além das informações adicionais usadas para verificação de erros no cabeçalho,

informações sobre fragmentação, e assim por diante. O resultado é o pacote da camada de rede, chamado de *datagrama*. A camada de rede passa, então, o pacote para a camada de enlace de dados.

4. A camada de enlace de dados trata o pacote da camada de rede como *payload* e adiciona o seu próprio cabeçalho, o qual contém os endereços da camada de enlace do *host* e do próximo salto (o roteador). O resultado é o pacote da camada de enlace, que é chamado de *quadro*. O quadro é passado para a camada física para transmissão.

Desencapsulamento e encapsulamento no roteador

No roteador, temos tanto desencapsulamento como encapsulamento, porque o roteador está ligado a dois ou mais enlaces.

1. Após o conjunto de *bits* ser entregue à camada de enlace de dados, esta desencapsula o datagrama a partir do quadro e passa tal datagrama para a camada de rede.
2. A camada de rede apenas inspeciona os endereços de origem e de destino no cabeçalho do datagrama e consulta sua tabela de roteamento para encontrar o próximo salto para o qual o datagrama deve ser entregue. O conteúdo do datagrama não deve ser alterado pela camada de rede no roteador a menos que seja necessário fragmentar o datagrama caso ele seja grande demais para passar pelo enlace seguinte. O datagrama é então passado para a camada de enlace de dados do enlace seguinte.
3. A camada de enlace de dados do enlace seguinte encapsula o datagrama em um quadro e passa o resultado para a camada física para transmissão.

Desencapsulamento no *host* de destino

No *host* de destino, cada camada apenas desencapsula o pacote recebido, remove o *payload* e entrega esta carga útil para o protocolo da camada imediatamente superior, até a mensagem chegar à camada de aplicação. É importante ressaltar que o desencapsulamento no *host* envolve a verificação de erros.

Endereçamento

Cabe mencionar outro conceito relacionado com protocolos em camadas na Internet: *endereçamento*. Como discutimos anteriormente, nesse modelo existe a comunicação lógica entre os pares nas camadas. Qualquer comunicação que envolve duas partes precisa de dois endereços: o endereço de origem e o endereço de destino. Embora pareça que precisamos de cinco pares de endereços, um par por camada, normalmente temos apenas quatro, porque a camada física não precisa de endereços; a unidade de dados trocados na camada física é o *bit*, o qual definitivamente não pode ter um endereço. A Figura 1.17 mostra o endereçamento em cada camada.

Figura 1.17 Endereçamento na pilha de protocolos TCP/IP.

Como mostra a figura, existe uma relação entre a camada, o endereço utilizado naquela camada e o nome do pacote nela. Na camada de aplicação, normalmente usamos nomes para definir o *site* que fornece os serviços, como *algumorg.com*, ou o endereço de e-mail, como *alguem@coldmail.com*. Na camada de transporte, os endereços são chamados de números de porta, e definem os programas da camada de aplicação na origem e no destino. Os números de porta são endereços locais que permitem fazer a distinção entre os vários programas sendo executados ao mesmo tempo. Na camada de rede, os endereços são globais, tendo toda a Internet como escopo. Um endereço da camada de rede define univocamente a conexão de um dispositivo com a Internet. Os endereços de camada de enlace, às vezes chamados endereços MAC, são endereços definidos localmente, cada um dos quais determina um *host* ou roteador específico em uma rede (LAN ou WAN). Falaremos mais sobre esses endereços em futuros capítulos.

Multiplexação e demultiplexação

Como a pilha de protocolos TCP/IP usa vários protocolos em algumas camadas, podemos dizer que temos multiplexação na origem e demultiplexação no destino. Multiplexação, neste caso, significa que um protocolo em uma camada pode encapsular um pacote de vários protocolos da camada imediatamente superior (um de cada vez); demultiplexação significa que um protocolo pode desencapsular e entregar um pacote para vários protocolos da camada imediatamente superior (um de cada vez). A Figura 1.18 mostra o conceito de multiplexação e de multiplexação nas três camadas mais altas.

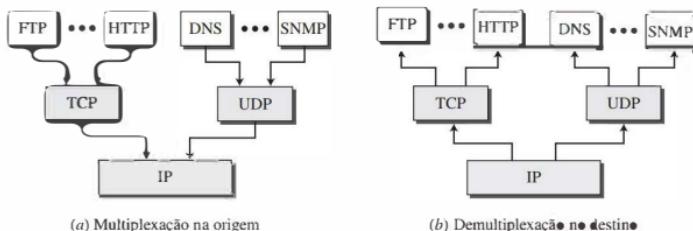



Figura 1.18 Multiplexação e demultiplexação.

Para ser capaz de multiplexar e demultiplexar, um protocolo precisa ter um campo em seu cabeçalho para identificar o protocolo ao qual os pacotes encapsulados pertencem. Na camada de transporte, tanto o UDP como o TCP podem aceitar mensagens de vários protocolos da camada de aplicação. Na camada de rede, o IP pode aceitar um segmento do TCP ou um datagrama de usuário do UDP. O IP pode também aceitar um pacote de outros protocolos, tais como ICMP, IGMP e assim por diante. Na camada de enlace de dados, um quadro pode carregar uma carga útil proveniente de protocolos IP ou outros protocolos, como ARP (ver Capítulo 5).

1.2.3 Modelo OSI

Embora, quando se fala da Internet, todo mundo discuta sobre a pilha de protocolos TCP/IP, esta não é a única pilha de protocolos existente. Estabelecida em 1947, a **Organização Internacional de Padronização** (ISO – International Organization for Standardization) é um órgão multinacional dedicado a criar normas internacionais aceitas mundialmente. Quase três quartos dos países do

mundo estão representados na ISO. Um padrão ISO que abrange todos os aspectos das redes de comunicação é o modelo **Interconexão de Sistemas Abertos** (OSI – Open Systems Interconnection). Ele foi introduzido pela primeira vez no final de 1970.

ISO é a organização; OSI é o modelo.

Um *sistema aberto* é um conjunto de protocolos que permite que quaisquer dois sistemas diferentes se comuniquem independentemente de suas arquiteturas subjacentes. O objetivo do modelo OSI é facilitar a comunicação entre sistemas diferentes sem a necessidade de mudanças na lógica do *hardware* e *software* subjacentes. O modelo OSI não é um protocolo, é um modelo para compreender e projetar uma arquitetura de rede que seja flexível, robusta e interoperável e foi criado com o objetivo de se tornar a base para a criação dos protocolos da pilha OSI.

O modelo OSI é uma estrutura em camadas para a concepção de sistemas de rede que permitam a comunicação entre todos os tipos de sistemas computacionais. Ele é constituído por sete camadas separadas, porém relacionadas, cada uma das quais definindo uma parte do processo de transferência de informação ao longo de uma rede (ver a Figura 1.19).

Figura 1.19 Modelo OSI.

OSI *versus* TCP/IP

Quando comparamos os dois modelos, percebemos que duas camadas, sessão e apresentação, estão ausentes na pilha de protocolos TCP/IP. Essas duas camadas não foram adicionados à pilha de protocolos TCP/IP após a publicação do modelo OSI. A camada de aplicação na pilha TCP/IP costuma ser considerada como a combinação de três camadas do modelo OSI, conforme mostra a Figura 1.20.

Dois motivos foram apresentados para tal decisão. Primeiro, o TCP/IP tem mais do que um protocolo da camada de transporte, e algumas das funcionalidades da camada de sessão estão disponíveis em alguns dos protocolos da camada de transporte. Em segundo lugar, a camada de aplicação não se resume a apenas um único programa, e muitos aplicativos podem ser desenvolvidas nesta camada. Se alguma das funcionalidades mencionadas nas camadas de sessão e apresentação for necessária para um aplicativo em particular, ela pode ser incluída no desenvolvimento daquele programa.

Falta de sucesso do modelo OSI

O modelo OSI surgiu depois da pilha de protocolos TCP/IP. Inicialmente, a maioria dos especialistas ficou entusiasmada e pensou que a pilha de protocolos TCP/IP seria totalmente substituída pelo modelo OSI. Isto não aconteceu por vários motivos, mas descrevemos apenas três, com os

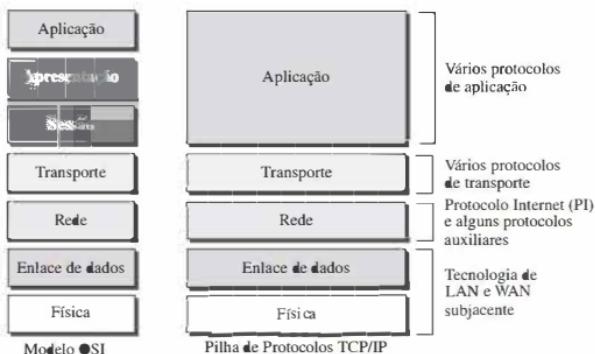



Figura 1.20 TCP/IP e Modelo OSI.

quais todos os especialistas da área concordam. Primeiro, o OSI foi concluído quando o TCP/IP já estava totalmente em vigor e muito tempo e dinheiro já haviam sido investidos nessa pilha de protocolos; mudar seria muito caro. Segundo, algumas camadas no modelo OSI nunca foram totalmente definidas. Por exemplo, embora os serviços providos pelas camadas de apresentação e de sessão fossem listados no documento, protocolos reais para essas duas camadas não foram totalmente definidos, nem descritos, e o *software* correspondente não foi totalmente desenvolvido. Terceiro, quando o OSI foi implementado por uma organização em uma aplicação diferente da Internet, o modelo não demonstrou um nível de desempenho elevado o suficiente para incentivar as autoridades responsáveis pela Internet a trocar a pilha de protocolos TCP/IP pelo modelo OSI.

1.3 HISTÓRIA DA INTERNET

Agora que proporcionamos uma visão geral da Internet e de seus protocolos, vamos apresentar uma breve história da Internet. Veremos como a Internet evoluiu de uma rede privada para uma rede global em menos de quarenta anos.

1.3.1 História inicial

Havia algumas redes de comunicação, tais como telégrafo e redes de telefonia, antes de 1960. Tais redes eram adequadas para comunicações a uma taxa constante daquela época, o que significa que, depois de gerar uma conexão entre dois usuários, as mensagens codificadas (telegrafia) ou de voz (telefonia) podiam ser trocadas. Uma rede de computadores, por outro lado, deve ser capaz de lidar com dados *em rajadas*, ou seja, dados recebidos a taxas variáveis em instantes distintos. O mundo precisava aguardar pela invenção da rede de comutação de pacotes.

Surgimento das redes de comutação de pacotes

A teoria da comutação de pacotes para tráfego em rajadas foi apresentada pela primeira vez por Leonard Kleinrock em 1961, no MIT. Ao mesmo tempo, dois outros pesquisadores, Paul Baran, do Rand Institute e Donald Davies, do National Physical Laboratory na Inglaterra, publicaram alguns trabalhos sobre redes de comutação de pacotes.

ARPANET

Em meados dos anos 1960, os computadores do tipo *mainframe* localizados em organizações de pesquisa eram dispositivos isolados. Computadores de diferentes fabricantes eram incapazes de se comunicar uns com os outros. A **Agência de Projetos de Pesquisa Avançados** (ARPA – Advanced Research Projects Agency), órgão do Departamento de Defesa (DOD – *Department of Defense*) dos Estados Unidos, estava interessada em encontrar uma maneira de conectar computadores para que os pesquisadores financiados por eles pudessem compartilhar suas descobertas, reduzindo custos e eliminando a duplicação de esforços.

Em 1967, em uma reunião da Associação para Maquinaria da Computação (ACM – Association for Computing Machinery), a ARPA apresentou suas ideias para a **Rede da Agência de Projetos de Pesquisa Avançados** (ARPANET – Advanced Research Projects Agency Network), uma pequena rede de computadores conectados. A ideia era que cada computador (não necessariamente do mesmo fabricante) seria ligado a um computador especializado, chamado de Processador de Mensagens de Interface (IMP – Interface Message Processor). Os IMPs, por sua vez, seriam ligados uns aos outros. Cada IMP precisava ser capaz de se comunicar com outros IMP, bem como com o *host* ao qual estava conectado.

Em 1969, a ARPANET tornou-se realidade. Quatro nós, na Universidade da Califórnia em Los Angeles (UCLA), na Universidade da Califórnia em Santa Barbara (UCSB), no Instituto de Pesquisa de Stanford (SRI) e na Universidade de Utah, foram conectados por meio de IMPs para formar uma rede. Um *software* chamado de Protocolo de Controle de Rede (NCP – Network Control Protocol), permitia a comunicação entre os *hosts*.

1.3.2 Surgimento da Internet

Em 1972, Vint Cerf e Bob Kahn, ambos parte do grupo que criou a ARPANET, colaboraram no que eles chamaram de *Internetting Project* (Projeto Inter-redes). Eles desejavam ligar redes distintas para que um *host* em uma rede pudesse se comunicar com um *host* em outra rede. Houve muitos problemas que precisaram ser superados: tamanhos de pacotes distintos, interfaces distintas, taxas de transmissão distintas, bem como requisitos de confiabilidade distintos. Cerf e Kahn tiveram a ideia de um dispositivo chamado de *gateway* para atuar como o *hardware* intermediário na transferência de dados de uma rede para outra.

TCP/IP

Em 1973, o artigo de Cerf e Kahn tornou-se um marco que esboçava os protocolos para se conseguir entrega de dados fim a fim. Era uma nova versão do NCP. Esse documento sobre protocolo de controle de transmissão (TCP – Transmission Control Protocol) incluía conceitos como encapsulamento, datagrama e as funções de um *gateway*. Uma ideia radical foi transferir a responsabilidade pela correção de erros, que passou do IMP para a máquina *host*. Esta Internet da ARPA tornou-se, então, o foco dos esforços de comunicação. Nessa mesma época, a responsabilidade sobre a ARPANET foi entregue à Agência de Comunicações de Defesa (DCA – Defense Communication Agency).

Em outubro de 1977, uma internet consistindo de três diferentes redes (ARPANET, Packet Radio e Packet Satellite) foi demonstrada com sucesso. A comunicação entre as redes era agora possível.

Pouco tempo depois, as autoridades decidiram dividir o TCP em dois protocolos: o TCP (Transmission Control Protocol) e o IP (Internet Protocol). O IP lidava com o roteamento de datagramas, enquanto o TCP ficava responsável por funções de mais alto nível, como segmentação, remontagem e detecção de erros. A nova combinação tornou-se conhecida como TCP/IP.

Em 1981, seguindo um contrato com o Departamento de Defesa americano, a UC Berkeley modificou o sistema operacional UNIX para incluir o TCP/IP. A inclusão deste *software* de rede junto a um sistema operacional popular teve grande influência no aumento da popularidade das redes. A implementação aberta (independente de um fabricante em específico) do UNIX Berkeley deu a todos os fabricantes um código-base funcional sobre o qual eles poderiam construir seus produtos.

Em 1983, autoridades aboliram os protocolos originais da ARPANET, e o TCP/IP tornou-se o protocolo oficial da ARPANET. Aqueles que desejassem usar a Internet para acessar um computador em uma rede diferente tinham que estar utilizando o TCP/IP.

MILNET

Em 1983, a ARPANET foi dividia em duas redes: a **Rede Militar** (MILNET – Military Network) para usuários militares e a ARPANET, para usuários não militares.

CSNET

Outro marco na história da Internet foi a criação da CSNET em 1981. A Rede de Ciência da Computação (CSNET – Computer Science Network) era uma rede patrocinada pela Fundação Nacional de Ciência (NSF – National Science Foundation). A rede foi concebida por universidades que eram inelegíveis para integrar a ARPANET devido à ausência de laços com o Departamento de Defesa dos Estados Unidos. A CSNET era uma rede mais barata; não apresentava enlaces redundantes e a taxa de transmissão era mais lenta.

Em meados dos anos 1980, a maioria das universidades dos Estados Unidos que tinham departamentos de ciência da computação fazia parte da CSNET. Outras instituições e empresas também estavam formando suas próprias redes e usando TCP/IP para interconexão. O termo *Internet*, originalmente associado a redes conectadas financiadas pelo governo, agora designava as redes conectadas usando protocolos TCP/IP.

NSFNET

Com o sucesso da CSNET, em 1986, a NSF patrocinou a Rede da **Fundação Nacional de Ciência** (NSFNET – National Science Foundation Network), um *backbone* que ligava cinco centros de supercomputadores localizados em diferentes regiões dos Estados Unidos. Redes comunitárias tiveram acesso a esse *backbone*, uma linha T-1 com uma taxa de transmissão de 1.544 Mbps, proporcionando, assim, conectividade em todo o território dos Estados Unidos. Em 1990, a ARPANET foi oficialmente aposentada e substituída pela NSFNET. Em 1995, a NSFNET foi revertida de volta ao seu conceito original de uma rede de pesquisa.

ANSNET

Em 1991, o governo dos Estados Unidos decidiu que a NSFNET não era capaz de suportar o tráfego da Internet que vinha aumentando rapidamente. Três empresas, IBM, Merit e Verizon, preencheram esta lacuna ao formar uma organização sem fins lucrativos chamada Serviços & Redes Avançados (ANS – Advanced Network & Services) para construir um novo *backbone* de Internet de alta velocidade chamado **Advanced Network Services Network** (ANSNET).

1.3.3 A Internet hoje

Hoje em dia, vemos um rápido crescimento tanto de infraestrutura como do surgimento de novas aplicações. A Internet, atualmente, é um conjunto de redes que prestam serviços para o mundo todo. O que tornou a Internet tão popular foi a invenção de novas aplicações.

World Wide Web

A década de 1990 testemunhou a explosão de aplicações de Internet devido ao surgimento da World Wide Web (WWW). A Web foi inventada no CERN por Tim Berners-Lee e foi adicionada às aplicações comerciais desenvolvidas para a Internet.

Multimídia

Desenvolvimentos recentes nas aplicações multimídia, tais como voz sobre IP (telefonia), vídeo sobre IP (Skype), compartilhamento de vídeos (YouTube), e televisão sobre IP (PPLive), aumentaram o número de usuários e o tempo que cada usuário gasta na rede. Discutiremos multimídia no Capítulo 8.

Aplicações *peer-to-peer*

Redes *peer-to-peer* (ou par a par) também constituem uma nova área de comunicação com grande potencial. Introduziremos algumas aplicações *peer-to-peer* no Capítulo 2.

1.4 PADRÕES E ADMINISTRAÇÃO

Na discussão sobre a Internet e seus protocolos, muitas vezes encontramos alguma referência sobre um padrão ou uma entidade administrativa. Nesta seção, apresentamos esses padrões e entidades administrativas para os leitores que não estão familiarizados com eles; esta seção pode ser ignorada se o leitor já estiver familiarizado com eles.

1.4.1 Padrões Internet

Um **padrão Internet** é uma especificação amplamente testada, útil e aceita por aqueles que trabalham com a Internet. É um conjunto de regras formais que devem ser seguidas. Existe um procedimento rigoroso para que uma especificação atinja o *status* de padrão Internet. A especificação começa como um Internet draft (um esboço do padrão). Um **Internet draft** é um documento de trabalho (um trabalho em andamento) sem qualquer *status* oficial e com uma vida útil de seis meses. Por recomendação das autoridades da Internet, esse esboço pode ser publicado como um **Pedido de Comentários** (RFC – Request for Comment). Cada RFC é editado, tem um número a ele atribuído e é disponibilizado a todos os interessados. Os RFCs passam por níveis de maturidade e são classificados de acordo com o seu nível de exigência.

Níveis de maturidade

Um RFC, durante seu tempo de vida, é classificado em um de seis *níveis de maturidade*: proposta de padrão, esboço de padrão, padrão Internet, histórico, experimental e informativo (ver Figura 1.21).

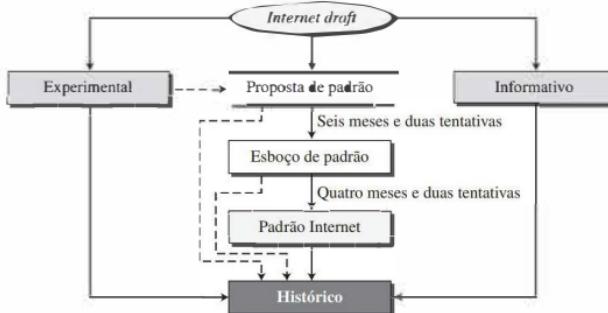



Figura 1.21 Níveis de maturidade de um RFC.

- **Proposta de padrão.** Uma proposta de padrão é uma especificação que está estável, bem compreendida e tem interesse suficiente da comunidade da Internet. Nesse nível, a especificação é geralmente testada e implementada por vários grupos diferentes.
- **Esboço de padrão.** A proposta de padrão é elevada ao *status* de esboço de padrão após pelo menos duas implementações independentes bem-sucedidas e interoperáveis. Salvo em caso de dificuldades, um esboço de padrão, com modificações em caso de problemas específicos serem encontrados, normalmente torna-se um padrão Internet.
- **Padrão Internet.** Um esboço de padrão atinge o *status* de padrão Internet após demonstrações bem-sucedidas de sua implementação.
- **Histórico.** Os RFCs históricos são significativos de um ponto de vista histórico. Eles podem ter sido substituídos por especificações posteriores ou nunca terem alcançado o nível de maturidade necessário para se tornarem um padrão Internet.
- **Experimental.** Um RFC classificado como experimental descreve trabalhos relacionados com situações experimentais que não afetam o funcionamento da Internet. Esse RFC não deveria ser implementado em qualquer serviço de Internet funcional.
- **Informativo.** Um RFC classificado como informativo contém informações gerais, histórico ou tutoriais relacionados à Internet. Geralmente, é escrito por alguém de uma organização que não pertence aos grupos que coordenam as questões da Internet, tal como um fornecedor de equipamentos.

Níveis de exigência

Os RFCs são classificados em cinco níveis de exigência: exigido, recomendado, opcional, uso limitado e não recomendado.

- **Exigido.** Um RFC é rotulado como *exigido* se tiver que ser implementado por todos os sistemas de Internet para que eles tenham conformidade mínima. Por exemplo, o IP (Capítulo 4) e o ICMP (Capítulo 4) são protocolos exigidos.
- **Recomendado.** Um RFC rotulado como recomendado não é necessário para se obter conformidade mínima; ele é recomendado devido a sua utilidade. Por exemplo, o FTP (Capítulo 2) e o TELNET (Capítulo 2) são protocolos recomendados.
- **Opcional.** Um RFC rotulado como *opcional* não é necessário nem recomendado. No entanto, um sistema pode usá-lo para o seu próprio benefício.
- **Uso limitado.** Um RFC rotulado como *uso limitado* deve ser usado somente em situações limitadas. A maioria dos RFCs experimentais se enquadra nessa categoria.
- **Não recomendado.** Um RFC rotulado como *não recomendado* é inadequado para uso geral. Normalmente, um RFC histórico (obsoleto) pode se enquadrar nessa categoria.

Os RFCs podem ser encontrados em http://www.rfc-editor.org.

1.4.2 Administração da Internet

A Internet, com suas raízes principalmente no domínio da pesquisa, evoluiu e ganhou uma base de usuários mais ampla com uma significativa atividade comercial. Vários grupos que coordenam as questões da Internet têm guiado seu crescimento e desenvolvimento. No Apêndice D,

disponível no *site* www.grupoa.com.br, podemos encontrar os endereços físicos, endereços de e-mail e números de telefone de alguns desses grupos. A Figura 1.22 mostra a organização geral da administração da Internet.

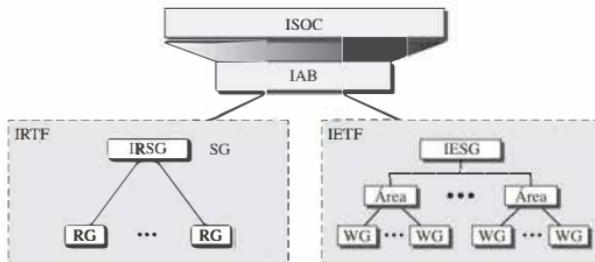



Figura 1.22 Níveis de maturidade de um RFC.

ISOC

A **Sociedade Internet** (ISOC – Internet Society) é uma organização internacional sem fins lucrativos formada em 1992 para fornecer apoio ao processo de criação de padrões Internet. A ISOC faz isso por meio da manutenção e apoio a outros órgãos administrativos da Internet, tais como IAB, IETF, IRTF e IANA (consulte as próximas seções). A ISOC também promove pesquisas e outras atividades acadêmicas relacionadas à Internet.

IAB

O **Conselho de Arquitetura da Internet** (IAB – Internet Architecture Board) é o órgão de consultoria técnica da ISOC. Os principais objetivos do IAB são supervisionar o desenvolvimento contínuo da pilha de protocolos TCP/IP e fornecer sua capacidade de assessoria técnica aos membros da comunidade de pesquisa da Internet. O IAB faz isso por meio dos seus dois componentes principais: a Força-Tarefa de Engenharia da Internet (IETF – Internet Engineering Task Force) e a Força-Tarefa de Pesquisa da Internet (IRTF – Internet Research Task Force). Outra responsabilidade do IAB é a gestão editorial dos RFCs, conforme descrito anteriormente. O IAB é também a ligação externa entre a Internet e outros fóruns e organizações de padronização.

IETF

A **Força-Tarefa de Engenharia da Internet** (IETF – Internet Engineering Task Force) é um fórum de grupos de trabalho gerenciados pelo Internet Engineering Steering Group (IESG). A IETF é responsável por identificar problemas operacionais e propor soluções para eles. A IETF também desenvolve e analisa as especificações que podem se tornar padrões Internet. Os grupos de trabalho são reunidos em áreas e cada área concentra-se em um tópico específico. Atualmente, nove áreas foram definidas, entre elas: aplicações, protocolos, roteamento, gerenciamento de rede de nova geração (IPng) e segurança.

IRTF

A **Força-Tarefa de Pesquisa da Internet** (IRTF – Internet Research Task Force) é um fórum de grupos de trabalho gerenciados pelo IRSG (Internet Research Steering Group). A IRTF concentra-se em temas de pesquisa de longo prazo relacionados com protocolos da Internet, aplicativos, arquitetura e tecnologia.

IANA e ICANN

A **Autoridade para Atribuição de Números na Internet** (IANA – Internet Assigned Numbers Authority), apoiada pelo governo dos Estados Unidos, era responsável pela gestão dos nomes de domínio e endereços na Internet até outubro de 1998. Naquela época, a Corporação para Atribuição de Nomes e Números na Internet (ICANN – Internet Corporation for Assigned Names and Numbers), uma corporação privada sem fins lucrativos gerenciada por um conselho internacional, assumiu as operações da IANA.

Núcleo de Informação e Coordenação

O **Núcleo de Informação e Coordenação** (NIC – Network Information Center) é responsável por coletar e distribuir informações sobre os protocolos TCP/IP.

Os endereços e *sites* de organizações da Internet podem ser encontrados no Apêndice D, no *site* do Grupo A.

1.5 MATERIAL DO FINAL DO CAPÍTULO

1.5.1 Leitura adicional

Para mais detalhes sobre os assuntos discutidos neste capítulo, recomendamos os seguintes livros, *sites*, e RFCs. Os itens entre colchetes referem-se à lista de referências no final do livro.

Livros e artigos

Diversos livros e artigos dão uma cobertura bastante completa da história da Internet, incluindo [Seg 98], [Lei *et al.* 98], [Kle 04], [Cer 89], e [Jen *et al.* 86].

RFCs

Dois RFCs em particular discutem a pilha TCP/IP: RFC 791 (IP) e RFC 817 (TCP). Em capítulos posteriores, listamos diferentes RFCs relacionadas a cada protocolo em cada camada.

1.5.2 Termos-chave

- Agência de Projetos de Pesquisa Avançados (ARPA – Advanced Research Projects Agency)
- Autoridade para Atribuição de Números na Internet (IANA – Internet Assigned Numbers Authority)
- Conselho de Arquitetura da Internet (IAB – Internet Architecture Board)
- Corporação para Atribuição de Nomes e Números na Internet (ICANN – Internet Corporation for Assigned Names and Numbers)
- Força-Tarefa de Engenharia da Internet (IETF – Internet Engineering Task Force)

- Força-Tarefa de Pesquisa da Internet (IRTF – Internet Research Task Force)
- internet
- Internet
- *internetwork*
- modelo OSI (Open Systems Interconnection)
- Núcleo de Informação e Coordenação (NIC – Network Information Center)
- Organização Internacional de Padronização (ISO – International Organization for Standardization)
- pilha de protocolos TCP/IP
- protocolo
- protocolos em camadas
- Provedor de Serviços de Internet (ISP – Internet Service Provider)
- Rede ANS (ANSNET – Advanced Network Services Network)
- Rede ARPA (ARPANET – Advanced Research Projects Agency Network)
- Rede da Fundação Nacional de Ciência (NSFNET – National Science Foundation Network)
- Rede de Ciência da Computação (CSNET – Computer Science Network)
- rede de comutação de circuitos
- rede de comutação de pacotes
- rede de longa distância (WAN – Wide Area Network)
- rede local (LAN – Local Area Network) Pedido de Comentários (RFC – Request for Comment)
- Rede Militar (MILNET – Military Network)
- Sociedade da Internet (ISOC – Internet Society)

1.5.3 Resumo

Uma rede é um conjunto de dispositivos conectados por enlaces de comunicação. Um dispositivo pode ser um computador, impressora ou qualquer outro dispositivo capaz de enviar e/ou receber dados gerados por outros nós da rede. Atualmente, quando falamos de redes, geralmente nos referimos a duas categorias principais: redes locais (LANs) e redes de longa distância (WANs). A Internet hoje é composta de muitas redes locais e de longa distância ligadas por dispositivos de conexão e estações de comutação. A maioria dos usuários finais que desejam conexão com a Internet utiliza os serviços de Provedores de Serviços de Internet (ISPs). Existem ISPs de *backbone*, ISPs regionais e ISPs locais.

Um protocolo é um conjunto de regras que governa a comunicação. Em protocolos em camadas, precisamos seguir dois princípios para prover comunicação bidirecional. Primeiro, cada camada deve realizar duas tarefas opostas. Segundo, dois objetos em cada camada em ambos os lados da comunicação devem ser idênticos. O TCP/IP é um conjunto de protocolos hierárquicos formado por cinco camadas: aplicação, transporte, rede, enlace de dados e física.

A história da interconexão de redes (*internetworking*) começou com a ARPA em meados dos anos 1960. O nascimento da Internet pode ser associado ao trabalho de Cerf e Kahn e à invenção de um *gateway* para conectar redes. A administração da Internet tem evoluído junto com a Internet. A ISOC promove pesquisa e atividades. O IAB provê consultoria técnica à ISOC. A IETF é um fórum de grupos de trabalho responsáveis por problemas operacionais. A IRTF é um fórum de grupos de trabalho com foco em temas de pesquisa de longo prazo. A ICANN é responsável pela gestão dos endereços e nomes de domínio na Internet. O NIC é responsável por coletar e distribuir informações sobre os protocolos TCP/IP.

Um padrão Internet é uma especificação completamente testada. Um *Internet draft* é um documento de trabalho sem *status* oficial e com uma vida útil de seis meses. Um *Internet draft* pode ser publicado como um RFC (*Request for Comment*). RFCs passam por níveis de maturidade e são classificados de acordo com seu nível de exigência.

1.6 ATIVIDADES PRÁTICAS

1.6.1 Testes

Diversos testes interativos relativos a este capítulo podem ser encontrados no *site* www.mhhe.com/forouzan. Recomendamos que o estudante realize os testes para verificar o seu entendimento dos materiais antes de continuar com as atividades práticas.

Questões

1-1 A transmissão em uma LAN com um cabo compartilhado (Figura 1.1*a*) é um exemplo de transmissão *broadcast* (um para muitos)? Explique.

1-2 Em uma LAN com um *switch* de camada de enlace (Figura 1.1*b*), o *Host* 1 deseja enviar uma mensagem para o *Host* 3. Dado que a comunicação acontece por meio do *switch* da camada de enlace, o *switch* precisa ter um endereço? Explique.

1-3 Quantas WANs ponto a ponto são necessárias para conectar *n* LANs se cada LAN deve ser capaz de se comunicar diretamente com qualquer outra LAN?

1-4 Quando usamos um telefone para falar com um amigo, estamos usando uma rede de comutação de circuitos ou uma rede de comutação de pacotes?

1-5 Quando uma pessoa usa uma conexão discada ou serviço DSL para se conectar à Internet, qual é o papel da companhia telefônica?

1-6 Qual é o primeiro princípio que discutimos neste capítulo sobre protocolo em camadas que deve ser seguido para fazer com que a comunicação seja bidirecional?

1-7 Quais camadas da pilha de protocolos TCP/IP estão envolvidas em um *switch* de camada de enlace?

1-8 Um roteador conecta três enlaces (redes). O roteador pode atuar em quantas de cada uma das camadas a seguir?

a. camada física
b. camada de enlace de dados
c. camada de rede

1-9 Na pilha de protocolos TCP/IP, quais são os objetos idênticos no lado do remetente e do destinatário quando consideramos a conexão lógica na camada de aplicação?

1-10 Um *host* se comunica com outro *host* usando a pilha de protocolos TCP/IP. Qual é a unidade de dados enviada ou recebida em cada uma das seguintes camadas?

1-11 Qual das seguintes unidades de dados é encapsulada em um quadro?

a. um datagrama de usuário
b. um datagrama
c. um segmento

1-12 Qual das seguintes unidades de dados é desencapsulada a partir de um datagrama do usuário?

a. um datagrama
b. um segmento
c. uma mensagem

1-13 Qual das seguintes unidades de dados contém uma mensagem da camada de aplicação mais o cabeçalho da camada 4?

a. um quadro
b. um datagrama de usuário
c. um bit

1-14 Liste alguns protocolos da camada de aplicação mencionados neste capítulo.

1-15 Se um número de porta tem 16 *bits* (2 *bytes*), qual é o tamanho mínimo do cabeçalho na camada de transporte da pilha de protocolos TCP/IP?

1-16 Quais são os tipos de endereços (identificadores) utilizados em cada uma das seguintes camadas?

a. camada de aplicação
b. camada de rede
c. camada de enlace de dados

1-17 Quando dizemos que a camada de transporte multiplexa e demultiplexa mensagens da camada de aplicação, queremos dizer que um protocolo de camada de transporte pode combinar várias mensagens da camada de aplicação em um único pacote? Explique.

1-18 Você sabe dizer por que não mencionamos os serviços de multiplexação/demultiplexação para a camada de aplicação?

1-19 Suponha que queiramos interconectar dois *hosts* isolados para que possam se comunicar entre si. Precisamos de um *switch* de camada de enlace entre os dois? Explique.

1-20 Se existe um único caminho entre o *host* de origem e o *host* de destino, precisamos de um roteador entre os dois?

1-21 Explique a diferença entre um *Internet draft* e uma proposta de padrão.

1-22 Explique a diferença entre um RFC exigido e um RFC recomendado.

1-23 Explique a diferença entre os papéis da IETF e da IRTF.

Problemas

1-1 Responda às seguintes questões sobre a Figura 1.10 quando a comunicação acontece de Maria para Ana:

a. Qual é o serviço provido pela camada 1 para a camada 2 no lado de Maria?

b. Qual é o serviço provido pela camada 1 para a camada 2 no lado de Ana?

1-2 Responda às seguintes questões sobre a Figura 1.10 quando a comunicação acontece de Maria para Ana:

a. Qual é o serviço provido pela camada 2 para a camada 3 no lado de Maria?

b. Qual é o serviço provido pela camada 2 para a camada 3 no lado de Ana?

1-3 Considere que o número de *hosts* conectados à Internet no ano de 2010 tenha sido 500 milhões. Se o número de *hosts* aumenta apenas 20% ao ano, qual será o número de *hosts* no ano de 2020?

1-4 Considere um sistema que utiliza cinco camadas de protocolo. Se o aplicativo cria uma mensagem de 100 *bytes* e cada camada (incluindo a quinta e a primeira) adiciona um cabeçalho de 10 *bytes* à unidade de dados, qual é a eficiência (a razão entre o número de *bytes* na camada de aplicação e o número de bytes transmitidos) do sistema?

1-5 Considere que criamos uma internet de comutação de pacotes. Usando a pilha de protocolos TCP/IP, precisamos transferir um arquivo enorme. Qual é a vantagem e a desvantagem de enviar pacotes grandes?

1-6 Ligue os seguintes conceitos a uma ou mais camadas da pilha de protocolos TCP/IP:

a. determinação de rota

b. conexão com o meio de transmissão

c. provimento de serviços para o usuário final

1-7 Ligue os seguintes conceitos a uma ou mais camadas da pilha de protocolos TCP/IP:

a. criação de datagramas de usuário

b. responsabilidade sobre o tratamento de quadros entre os nós adjacentes

c. transformação de *bits* em sinais eletromagnéticos

1-8 Na Figura 1.18, quando o protocolo IP desencapsula o pacote da camada de transporte, como ele sabe a qual protocolo da camada superior (UDP ou TCP) o pacote deve ser entregue?

1-9 Considere que uma internet privada use três protocolos diferentes na camada de enlace de dados (L1, L2 e L3). Redesenhe a Figura 1.18 com base nessa suposição. Podemos dizer que, na camada de enlace de dados, temos demultiplexação no nó de origem e demultiplexação no nó de destino?

1-10 Considere que uma internet privada exija que as mensagens da camada de aplicação sejam cifradas e decifradas por razões de segurança. Se precisarmos adicionar alguma informação sobre o processo de cifração/decifração (tais como os algoritmos utilizados no processo), isto significa que estamos adicionando uma camada à pilha de protocolos TCP/IP? Redesenhe as camadas do TCP/IP (Figura 1.12*b*) se você acredita que sim.

1-11 Protocolos em camadas podem ser encontrados em muitos aspectos de nossas vidas, tais como viagens aéreas. Imagine que você faça uma viagem de ida e volta para passar algum tempo em férias em um hotel. Você precisa passar por alguns processos no aeroporto da sua cidade antes de tomar o avião. Também precisa passar por alguns processos quando chegar ao aeroporto do *resort*. Mostre as camadas de protocolo para a viagem de ida e volta com algumas camadas, tais como *check-in*/recuperacão de bagagem, embarque/desembarque, decolagem/pouso.

1-12 Na Figura 1.4 no texto, existe um único caminho entre um *host* no escritório da costa oeste e um *host* no escritório da costa leste. Por que precisamos de dois roteadores nesta *internetwork*?

1-13 A apresentação dos dados está se tornando cada vez mais importante na Internet atual. Algumas pessoas argumentam que a pilha de protocolos TCP/IP precisa adicionar uma nova camada para cuidar da apresentação dos dados (ver Anexo C). Se esta nova camada for adicionada no futuro, qual deve ser a sua posição na pilha? Redesenhar a Figura 1.12 para incluir a camada.

1-14 Em uma internet, trocamos a tecnologia de LAN para uma nova tecnologia. Quais camadas

da pilha de protocolos TCP/IP precisam ser alteradas?

1-15 Suponha que um protocolo da camada de aplicação seja escrito para usar os serviços do UDP. Este protocolo da camada de aplicação pode passar a utilizar os serviços do TCP sem alterações?

Usando a internet da Figura 1.4 no texto, mostre as camadas da pilha de protocolos TCP/IP e do fluxo de dados quando dois *hosts*, um na costa oeste e outro na costa leste, trocam mensagens entre si.

1.7 EXPERIMENTOS DE SIMULAÇÃO

1.7.1 *Applets*

Uma das maneiras de mostrar os protocolos de rede em ação ou observar a solução de alguns exemplos é através do uso de animações interativas. Criamos alguns *applets* Java para mostrar alguns dos principais conceitos discutidos neste capítulo. Recomendamos que o estudante ative esses *applets* no *site* www.mhhe.com/forouzan e examine cuidadosamente os protocolos em ação.

1.7.2 Experimentos de laboratório

Experiências com redes e equipamentos de rede podem ser feitas usando pelo menos dois métodos. No primeiro método, podemos criar um laboratório de redes isoladas e utilizar *hardware* e *software* de rede para simular os tópicos abordados em cada capítulo. Podemos criar uma internet e enviar e receber pacotes de um *host* qualquer para outro. O fluxo de pacotes pode ser observado e o desempenho, medido. Embora esse método seja mais eficaz e mais instrutivo do que o segundo, seu custo de implementação é mais elevado e nem todas as instituições estão prontas para investir em tal laboratório exclusivo.

No segundo método, podemos usar a Internet, a maior rede no mundo, como nosso laboratório virtual; podemos enviar e receber pacotes através dela. A existência de algum *software* gratuito disponível para *download* nos permite capturar e examinar os pacotes trocados. Podemos analisá-los para ver como os aspectos teóricos das redes são colocados em ação. Embora o segundo método possa não ser tão eficaz como o primeiro, já que não podemos controlar e modificar os percursos dos pacotes para ver como a Internet se comporta, esse método é muito mais barato de ser implementado, pois não exige um laboratório de rede físico; ele pode ser implementado usando nosso desktop ou laptop. O *software* necessário é também gratuito e está disponível para *download*.

Existem muitos programas e utilitários disponíveis para os sistemas Windows e UNIX, que nos permitem monitorar, capturar, rastrear e analisar os pacotes que são trocados entre o nosso computador e a Internet. Alguns deles, como o *Wireshark* e o *Ping Plotter*, têm uma Interface Gráfica de Usuário (GUI – Graphical User Interface); outros, como *traceroute*, *nslookup*, *dig*, *ipconfig* e *ifconfig* são utilitários de linha de comando para administração de redes. Qualquer um desses programas e utilitários pode ser uma ferramenta de *debugging* valiosa para administradores de rede e uma ferramenta educacional para estudantes de rede de computadores.

Neste livro, usamos principalmente o Wireshark para experimentos de laboratório, embora ocasionalmente usemos outras ferramentas. O Wireshark captura, em tempo real, pacotes de dados chegando a uma interface de rede e então os exibe com as informações detalhadas do protocolo utilizado. O Wireshark, no entanto, é um analisador passivo. Ele apenas "mede" os dados da rede sem manipulá-los; não injeta pacotes na rede nem faz outras operações ativas. O Wireshark também não é uma ferramenta para detecção de invasões. Ele não emite avisos sobre qualquer invasão na rede, mas pode ajudar os administradores ou engenheiros de segurança de rede a descobrir o que está acontecendo dentro de uma rede e a solucionar problemas. Além de ser indispensável para administradores de rede e engenheiros de segurança, o Wireshark é uma ferramenta valiosa

para desenvolvedores de protocolo, que podem utilizá-lo para depurar implementações de protocolos, e um ótimo recurso educacional para estudantes de redes de computadores, que podem usá-lo para ver os detalhes operacionais de protocolos em tempo real.

Neste experimento de laboratório, aprendemos como baixar e instalar o Wireshark; as instruções estão disponíveis no *site* www.grupoa.com.br. Neste documento, também discutimos a ideia geral por trás do *software*, o formato de sua janela, e como usá-lo. O estudo completo desse laboratório prepara o estudante para usar o Wireshark em seus experimentos de laboratório para o próximo capítulo.

CAPÍTULO

2 CAMADA DE APLICAÇÃO

A Internet inteira, tanto *hardware* como *software*, foi projetada e desenvolvida para fornecer serviços na camada de aplicação. É na quinta camada da pilha de protocolos TCP/IP que esses serviços são fornecidos para os usuários da Internet; as outras quatro camadas existem para torná-los possíveis. Uma forma de estudar a tecnologia da Internet é primeiramente explicar os serviços prestados na camada de aplicação e, em seguida, mostrar como as outras quatro camadas dão suporte a esses serviços. Como esta é a abordagem que adotamos neste livro, a camada de aplicação será a primeira a ser abordada.

Desde a criação da Internet, muitos protocolos de aplicação foram criados e usados. Algumas dessas aplicações foram projetadas para algum uso específico e nunca tornaram-se padrões; algumas tornaram-se obsoletas. Outras foram modificadas ou substituídas por novas soluções e várias sobreviveram e se tornaram aplicações padronizadas. Novos protocolos de aplicação são constantemente adicionados à Internet.

Neste capítulo, a camada de aplicação será discutida em cinco seções.

- Na primeira seção, apresentamos a natureza dos serviços prestados pela Internet e duas categorias de aplicações: a tradicional, com base no *paradigma cliente-servidor*, e a mais nova, com base no *paradigma peer-to-peer*.
- Na segunda seção, discutimos o conceito do paradigma cliente-servidor e como ele fornece serviços para os usuários da Internet.
- Na terceira seção, descrevemos algumas aplicações predefinidas ou padrões com base no paradigma cliente-servidor. Discutimos também algumas aplicações populares, como navegação na Web, transferência de arquivos, e-mail e assim por diante.
- Na quarta seção, discutimos o conceito e os protocolos do paradigma *peer-to-peer*. Apresentamos alguns protocolos como Chord, Pastry, e Kademlia e citamos alguns aplicativos populares que utilizam tais protocolos.
- Na quinta seção, mostramos como uma nova aplicação pode ser criada usando paradigma cliente-servidor escrevendo dois programas na linguagem C, um para um cliente e outro para um servidor. No Capítulo 11, mostraremos como podemos escrever programas cliente-servidor na linguagem Java.

2.1 INTRODUÇÃO

A camada de aplicação provê serviços para o usuário. A comunicação é fornecida através de uma conexão lógica, o que significa que as duas camadas de aplicação consideram a existência de uma conexão direta imaginária por meio da qual podem enviar e receber mensagens. A Figura 2.1 ilustra a ideia por trás dessa conexão lógica.

A figura mostra uma situação na qual um cientista de uma empresa de pesquisa, chamada Pesquisa Celeste, precisa encomendar um livro relacionado à sua pesquisa em uma livraria

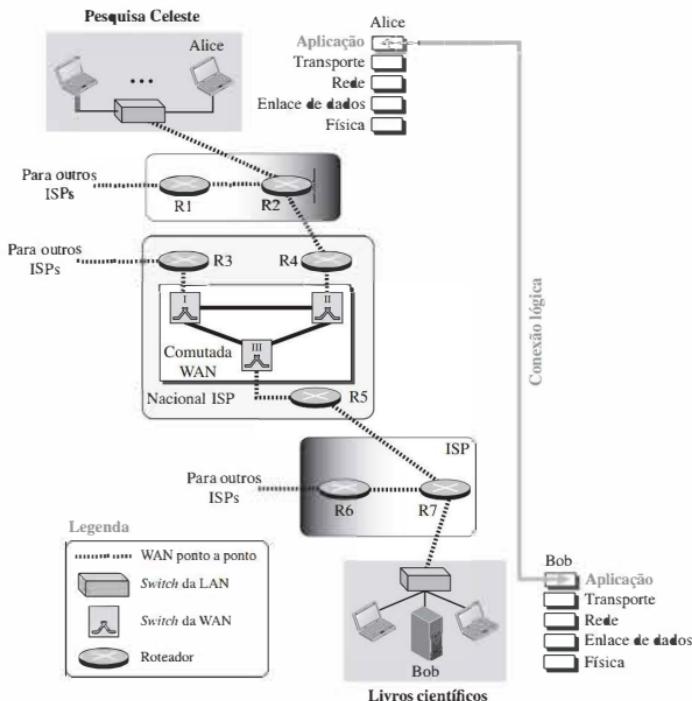



Figura 2.1 Conexão lógica na camada de aplicação.

online, denominada Livros Científicos. A conexão lógica se dá entre a camada de aplicação em um computador na Pesquisa Celeste e a de um servidor na Livros Científicos. Denominamos a primeira estação de Alice, e a segunda, de Bob. A comunicação na camada de aplicação é lógica, não física. Alice e Bob consideram que existe um canal lógico bidirecional entre eles, pelo qual podem enviar e receber mensagens. A comunicação real, no entanto, dá-se através de vários dispositivos (Alice, R2, R4, R5, R7 e Bob) e canais físicos, conforme mostrado na figura.

2.1.1 Fornecendo serviços

Todas as redes de comunicação que surgiram antes da Internet foram concebidas para oferecer serviços aos usuários da rede. A maioria delas, no entanto, foi originalmente projetada para um serviço específico. Por exemplo, a rede telefônica foi concebida para o serviço de voz: o objetivo era permitir que pessoas do mundo todo falassem umas com as outras. Mais tarde, no entanto, ela foi utilizada para prover outros serviços, como fac-símile (fax), habilitado por usuários que adicionavam algum *hardware* extra em ambas as extremidades da comunicação.

A Internet foi originalmente concebida com a mesma finalidade: fornecer o serviço para usuários ao redor do mundo. A arquitetura em camadas da pilha de protocolos TCP/IP, no entanto, torna a Internet mais flexível do que outras redes de comunicação, como redes de correios ou de telefonia. Cada camada da pilha foi criada com um ou mais protocolos, mas novos protocolos podem ser adicionados e outros podem ser removidos ou substituídos pelas autoridades da Internet. No entanto, se um protocolo é adicionado em qualquer camada, ele deve ser projetado de maneira a usar o serviço fornecido por um dos protocolos da camada inferior. Se um protocolo for removido de uma camada, deve-se ter o cuidado de alterar o protocolo da camada imediatamente superior que porventura use o serviço do protocolo removido.

A camada de aplicação, no entanto, é de certa forma diferente das outras camadas, pois é a camada mais alta da pilha. Os protocolos nessa camada não fornecem serviços a outros da pilha; apenas recebem os serviços dos protocolos da camada de transporte. Isto significa que os protocolos dessa camada podem ser facilmente removidos. Novos protocolos também podem ser adicionados, contanto que sejam capazes de utilizar o serviço fornecido por outro protocolo da camada de transporte.

Como a camada de aplicação é a única que fornece serviços para os usuários da Internet, sua flexibilidade, conforme descrito anteriormente, permite que novos protocolos de aplicação sejam adicionados à Internet, o que vem ocorrendo desde sua criação. Quando a Internet surgiu, havia poucos protocolos de aplicação disponíveis aos usuários; hoje, somos incapazes de enumerar esses protocolos porque novas soluções são constantemente adicionadas.

Protocolos padronizados e não padronizados

Para proporcionar um bom funcionamento da Internet, os protocolos utilizados nas primeiras quatro camadas da pilha TCP/IP precisam ser padronizados e documentados. Eles normalmente se tornam parte dos pacotes que vêm inclusos em sistemas operacionais, como Windows ou UNIX. Para que haja maior flexibilidade, entretanto, os protocolos da camada de aplicação podem ser tanto padronizados como não padronizados.

Protocolos padronizados da camada de aplicação

Existem diversos protocolos da camada de aplicação que foram padronizados e documentados pelas autoridades da Internet, e são utilizados em nossa interação diária com a rede. Cada protocolo-padrão consiste em um par de programas de computador que interagem com o usuário e com a camada de transporte para fornecer um serviço específico. Discutiremos alguns desses aplicativos padronizados mais adiante neste capítulo; alguns serão discutidos em outros capítulos. No estudo desses protocolos de aplicação, precisamos saber quais tipos de serviços eles fornecem, como funcionam, as opções que temos ao usar os aplicativos, etc. O estudo dos protocolos permite que um gerente de rede resolva facilmente possíveis problemas na utilização deles. Com esse conhecimento, também poderemos criar novos protocolos não padronizados.

Protocolos não padronizados da camada de aplicação

Um programador pode criar um protocolo não padronizado para a camada de aplicação se for capaz de escrever dois programas que prestem serviços para o usuário por meio da interação com a camada de transporte. Mais adiante neste capítulo, mostraremos como podemos escrever esses tipos de programas. A criação de um protocolo não padronizado, que não precisa sequer da aprovação das autoridades da Internet para ser usado de modo privado, é o que fez a Internet tornar-se tão popular mundialmente. Uma empresa privada pode criar um novo protocolo personalizado da camada de aplicação para se comunicar com todos os seus escritórios ao redor do mundo usando os serviços fornecidos pelas primeiras quatro camadas da pilha de protocolos TCP/IP, sem usar qualquer dos programas de aplicação padronizados. Para isto, basta escrever programas, usando alguma das linguagens de programação que usem os serviços fornecidos pelos protocolos da camada de transporte.

2.1.2 Paradigmas da camada de aplicação

Deve ficar claro que, para usar a Internet, precisamos de dois programas que interagem entre si: cada um sendo executado em um computador em algum lugar do mundo. Os dois programas precisam enviar mensagens um ao outro por meio da infraestrutura da Internet. No entanto, não discutimos qual relação deve existir entre esses programas. Ambos os aplicativos devem ser capazes de solicitar e prestar serviços ou cada um deve desempenhar apenas uma dessas funções? Dois paradigmas foram desenvolvidos desde o surgimento da Internet para responder a essa pergunta: o *paradigma cliente-servidor* e o *paradigma peer-to-peer*. Apresentaremos brevemente os dois nesta seção, mas discutiremos cada um deles com mais detalhes mais adiante.

Paradigma tradicional: cliente-servidor

O paradigma tradicional é chamado **paradigma cliente-servidor**, que era o mais popular até poucos anos atrás. Nele, o provedor de serviços é um aplicativo chamado processo-servidor; é executado continuamente à espera de outro aplicativo, chamado processo-cliente, que cria uma conexão através da Internet e solicita o serviço. Normalmente, existem alguns processos-servidores que podem fornecer um tipo específico de serviço, mas há muitos clientes que solicitam o serviço de qualquer um desses processos-servidores. O processo-servidor deve ser executado o tempo todo; o processo-cliente é iniciado quando o cliente precisa receber o serviço.

O paradigma cliente-servidor é semelhante a alguns serviços disponíveis fora do domínio da Internet. Por exemplo, uma central telefônica em qualquer área pode ser vista como um servidor; um assinante que liga e pede um número de telefone específico para a central pode ser visto como um cliente. A central deve estar pronta e disponível o tempo todo; o assinante pode ligar por um curto intervalo, apenas quando o serviço for necessário.

Apesar de a comunicação no paradigma cliente-servidor se dar entre dois aplicativos, o papel de cada programa é totalmente diferente. Em outras palavras, não podemos executar o programa-cliente como um programa-servidor ou vice-versa. Mais adiante neste capítulo, na explicação sobre programação no paradigma cliente-servidor, mostraremos que sempre precisamos escrever dois programas aplicativos para cada tipo de serviço. A Figura 2.2 mostra um exemplo da comunicação cliente-servidor, na qual três clientes se comunicam com um servidor para receber os serviços fornecidos por ele.

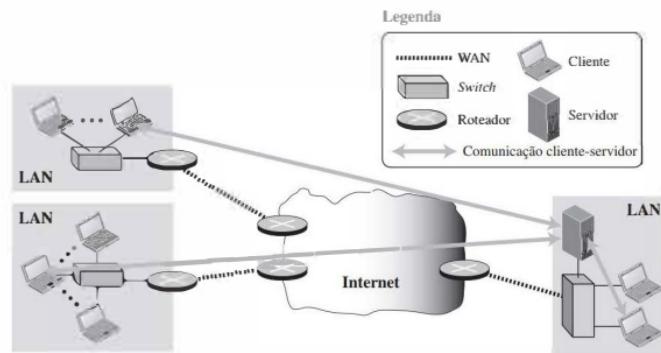



Figura 2.2 Exemplo do paradigma cliente-servidor.

Um problema com esse paradigma é que a carga de comunicação concentra-se no lado do servidor, o que significa que este deve ser um computador poderoso. Porém, mesmo um computador poderoso pode acabar sobrecarregado se um grande número de clientes tentar se conectar ao servidor ao mesmo tempo. Outro problema é que deve haver um provedor de serviços disposto a criar um servidor poderoso para um serviço específico e arcar com os custos envolvidos, o que significa que o serviço deve sempre gerar algum tipo de receita para o servidor de modo a incentivar tal empreendimento.

Vários serviços tradicionais ainda usam esse paradigma, incluindo o World Wide Web (WWW) e seu instrumento de acesso, o Protocolo de Transferência de Hipertexto (HTTP – HyperText Transfer Protocol), o Protocolo de Transferência de Arquivos (FTP – File Transfer Protocol), o Secure Shell (SSH), o e-mail e diversos outros. Discutiremos alguns desses protocolos e aplicativos mais adiante.

Novo paradigma: *peer-to-peer*

Um novo paradigma, chamado **paradigma *peer-to-peer*** ou **paradigma par a par** (geralmente abreviado como *paradigma P2P*) surgiu para responder às necessidades de novas aplicações. Nesse paradigma, não há necessidade de um processo-servidor que seja executado o tempo todo e espere pela conexão dos processos clientes. A responsabilidade é compartilhada entre os *peers* (os nós da rede). Um computador conectado à Internet pode prover serviços em dado momento e receber serviços em outro; pode até mesmo prover e receber serviços ao mesmo tempo. A Figura 2.3 mostra um exemplo da comunicação nesse paradigma.

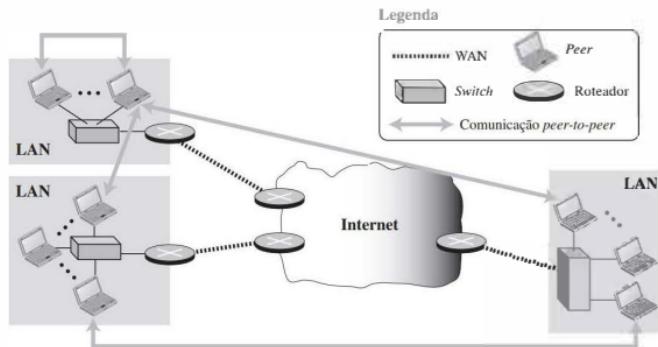



Figura 2.3 Exemplo do paradigma *peer-to-peer*.

Uma das áreas que realmente se encaixa nesse paradigma é a de telefonia via Internet. A comunicação por telefone é de fato uma atividade *peer-to-peer*; nenhuma das partes precisa executar o tempo todo à espera de uma chamada da outra. Outra área na qual o paradigma *peer-to-peer* pode ser usado é quando alguns computadores conectados à Internet têm algo para compartilhar uns com os outros. Por exemplo, se um usuário da Internet tem um arquivo disponível para compartilhar com outros usuários, não é preciso que o dono do arquivo se torne um servidor e execute um processo-servidor o tempo todo esperando que outros usuários se conectem e recuperem o arquivo.

Embora o paradigma *peer-to-peer* apresente elevada escalabilidade e seja comprovadamente eficaz em eliminar a necessidade de servidores caros precisarem ser executados e consertados o

tempo todo, ele apresenta também alguns desafios. O principal deles refere-se à segurança, pois é mais difícil criar uma comunicação segura entre serviços distribuídos do que entre serviços controlados por alguns servidores dedicados. Outro desafio é sua aplicabilidade, pois aparentemente nem todas as aplicações são capazes de utilizar esse novo paradigma. Por exemplo, não haveria muitos usuários da Internet prontos para se envolver caso um dia a Web fosse implementada como um serviço *peer-to-peer*.

Existem alguns novos aplicativos, tais como BitTorrent, Skype, IPTV e telefonia por Internet, que adotam esse paradigma. Discutiremos alguns deles neste capítulo e deixaremos a discussão sobre alguns outros aplicativos para capítulos subsequentes.

Paradigma misto

Uma aplicação pode optar por uma mistura dos dois paradigmas, combinando as vantagens de ambos. Por exemplo, uma comunicação cliente-servidor envolvendo o tráfego de uma quantidade pequena de dados pode ser usada para encontrar o endereço do *peer* que fornece um serviço. Após o endereço do *peer* ser encontrado, o serviço em si pode ser recebido daquele *peer* usando o paradigma *peer-to-peer*.

2.2 PARADIGMA CLIENTE-SERVIDOR

Em um paradigma cliente-servidor, a comunicação na camada de aplicação se dá entre dois aplicativos em execução denominados **processos**: um *cliente* e um *servidor*. Um cliente é um programa em execução que inicia a comunicação enviando um pedido (ou solicitação); um servidor é outro programa que aguarda pedidos de clientes. O servidor trata o pedido recebido, prepara um resultado e envia o resultado de volta ao cliente. Nessa definição, um servidor deve estar sendo executado quando o pedido de um cliente chegar, mas o cliente pode ser executado somente quando necessário. Ou seja, se tivermos dois computadores conectados um ao outro em algum lugar, podemos executar o processo-cliente em um deles e o processo-servidor no outro. No entanto, precisamos ter cuidado para que o programa-servidor seja iniciado antes do início da execução do programa-cliente. Em outras palavras, o tempo de vida de um servidor é infinito: ele deve ser iniciado e então executar para sempre, à espera dos clientes. O tempo de vida de um cliente é finito: ele normalmente envia um número finito de solicitações ao servidor correspondente, recebe as respostas e encerra.

2.2.1 Interface de programação de aplicativos

Como um processo-cliente pode se comunicar com um processo-servidor? Um programa de computador costuma ser escrito em uma linguagem de programação com um conjunto predefinido de instruções que dizem ao computador o que fazer. Uma linguagem de programação tem um conjunto de instruções para operações matemáticas, outro para a manipulação de cadeias de caracteres, mais um para acesso a dispositivos de entrada/saída e assim por diante. Se precisamos que um processo seja capaz de se comunicar com outro, usaremos um novo conjunto de instruções para dizer às quatro camadas mais baixas da pilha de protocolos TCP/IP que abram a conexão, enviem os dados à outra estação, recebam os dados dela e fechem a conexão. Um conjunto de instruções desse tipo é normalmente chamado **Interface de Programação de Aplicativos** (API – Application Programming Interface). Uma interface de programação é um conjunto de instruções entre duas entidades. Nesse caso, uma das entidades é o processo na camada de aplicação e a outra é o *sistema operacional* que encapsula as quatro primeiras camadas da pilha de protocolos TCP/IP. Em outras palavras, um fabricante de computadores precisa construir as quatro primeiras camadas da pilha no sistema operacional e incluir uma API; assim, os processos em execução na camada de aplicação são capazes de se comunicar com o sistema operacional ao enviar e receber mensagens pela Internet. Várias APIs foram projetadas para comunicação. Três delas são bastante comuns: **interface *socket***,

Interface de Camada de Transporte (TLI – Transport Layer Interface), e **STREAM**. Nesta seção, discutiremos brevemente apenas a *interface socket*, que é a mais comum, para dar uma ideia geral da comunicação em rede na camada de aplicação.

A interface *socket* surgiu no início de 1980 na Universidade de Berkeley como parte de um ambiente UNIX; um conjunto de instruções que fornecem comunicação entre a camada de aplicação e o sistema operacional, como mostrado na Figura 2.4. É um conjunto de instruções que podem ser usadas por um processo para se comunicar com outro.

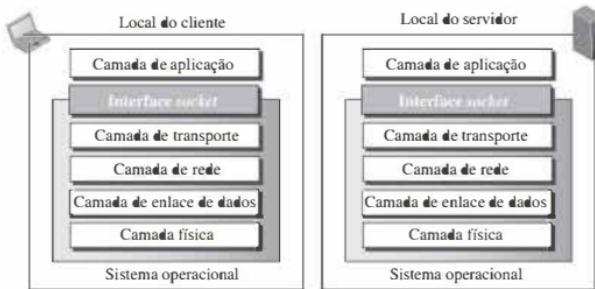



Figura 2.4 Posição da interface *socket*.

A ideia de *sockets* nos permite usar o conjunto de todas as instruções já projetadas em uma linguagem de programação para outros dispositivos de entrada e saída. Por exemplo, na maioria das linguagens de programação, como C, C++ ou Java, existem várias instruções para ler e gravar dados em dispositivos de entrada e saída, como um teclado (entrada), um monitor (saída), ou um arquivo (entrada e saída). Podemos usar as mesmas instruções para ler ou escrever em *sockets*, ou seja, estamos adicionando apenas novas entidades de entrada e saída à linguagem de programação, sem alterar a forma de enviar ou receber dados. A Figura 2.5 ilustra essa ideia e compara os *sockets* com outras entidades de entrada e saída.

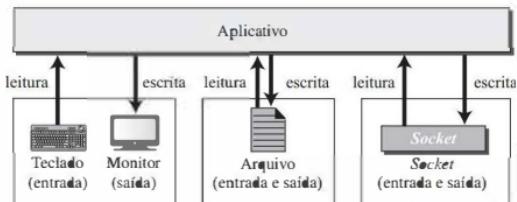



Figura 2.5 *Sockets* usados da mesma forma que outras entidades de entrada e saída.

Sockets

Apesar de um *socket* ter que se comportar como um terminal ou um arquivo, ele não é uma entidade física como eles, mas sim, uma abstração; é uma *estrutura de dados* criada e utilizada pelo aplicativo.

Pode-se dizer que, no que diz respeito à camada de aplicação, a comunicação entre um processo-cliente e um processo-servidor se limita à comunicação entre dois *sockets*, criados nos dois sistemas finais, como mostrado na Figura 2.6. O cliente percebe o *socket* como a entidade que recebe o pedido e envia a resposta; o servidor percebe o *socket* como aquele que tem um pedido e precisa da resposta. Se criarmos dois *sockets*, um em cada sistema final, e definirmos os endereços de origem e destino corretamente, podemos usar as instruções disponíveis para enviar e receber dados. O resto é responsabilidade do sistema operacional e do protocolo TCP/IP a ele incorporado.

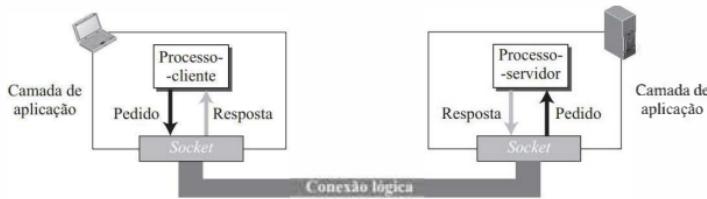



Figura 2.6 Uso de *sockets* na comunicação processo a processo.

Endereços de *socket*

A interação entre um cliente e um servidor é uma comunicação bidirecional, na qual precisamos de um par de endereços: um local (remetente) e um remoto (receptor). O endereço local em uma direção é o endereço remoto na outra direção e vice-versa. Como a comunicação no paradigma cliente-servidor é entre dois *sockets*, precisamos de um par de **endereços de *socket*** para a comunicação: um endereço de *socket* local e outro de *socket* remoto. No entanto, devemos definir um endereço de *socket* em termos de identificadores utilizados na pilha de protocolos TCP/IP.

Um endereço de *socket* deve primeiro definir o computador no qual um cliente ou um servidor está sendo executado. Como discutiremos no Capítulo 4, um computador na Internet é exclusivamente definido pelo seu endereço IP, um número inteiro de 32 *bits* na versão atual da rede. No entanto, vários processos cliente e servidor podem estar em execução ao mesmo tempo em um computador, o que significa que precisamos de um outro identificador para definir o processo-cliente ou servidor específico envolvido na comunicação. Como discutiremos no Capítulo 3, um aplicativo pode ser definido por um número de porta, um número inteiro de 16 *bits*. Isto significa que um endereço de *socket* deve ser uma combinação de um endereço IP e um número de porta, conforme mostrado na Figura 2.7.

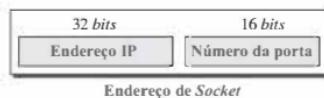



Figura 2.7 Endereço de *socket*.

Como um *socket* define o ponto final da comunicação, podemos dizer que um *socket* é identificado por um par de endereços de *socket*, um local e um remoto.

Exemplo 2.1

Podemos encontrar um endereço composto por dois níveis na comunicação telefônica. Um número de telefone pode definir uma organização, e, dentro dela, um ramal pode definir uma conexão especifica. O número de telefone, nesse caso, é similar ao endereço IP, que define a organização como um todo; o ramal é semelhante ao número de porta, que define a conexão específica.

Descobrindo endereços de *socket*

Como um cliente ou um servidor descobrem um par de endereços de *socket* para a comunicação? A situação é diferente em cada um dos lados.

Lado do servidor

O servidor precisa de um endereço de *socket* local (servidor) e de um endereço de *socket* remoto (cliente) para a comunicação.

Endereço de *socket* local O endereço de *socket* local (servidor) é fornecido pelo sistema operacional. O sistema operacional sabe o endereço IP do computador no qual o processo-servidor está sendo executado. O número da porta de um processo-servidor, no entanto, precisa ser atribuído. Se o processo-servidor for um processo padronizado, definido pelas autoridades da Internet, já existe um número de porta atribuído a ele. Por exemplo, o número de porta atribuído para o Protocolo de Transferência de Hipertexto (HTTP – Protocolo de Transferência de Hipertexto) é o número inteiro 80, que não pode ser utilizado por outro processo. Discutiremos esses números de porta bastante conhecidos no Capítulo 3. Se o processo-servidor não for padronizado, o projetista do processo-servidor pode escolher um número de porta, na faixa definida pelas autoridades da Internet, e atribuí-lo ao processo. Quando um servidor começa a ser executado, ele sabe o endereço de *socket* local.

Endereço de *socket* remoto O endereço de *socket* remoto para um servidor é o endereço de *socket* do cliente que inicia a conexão. Como o servidor pode atender muitos clientes, ele não sabe de antemão o endereço de *socket* remoto para a comunicação. O servidor pode encontrar o endereço quando um cliente tenta se conectar ao servidor. O endereço de *socket* do cliente, que está contido no pacote de solicitação enviado para o servidor, torna-se o endereço de *socket* remoto usado para responder àquele cliente. Em outras palavras, embora o endereço de *socket* local para um servidor seja fixo e utilizado durante todo seu tempo de vida, o remoto é alterado em cada interação com um cliente diferente.

Lado do cliente

O cliente também precisa de um endereço de *socket* local (cliente) e de um endereço de *socket* remoto (servidor) para a comunicação.

Endereço de *socket* local O endereço de *socket* local (cliente) também é fornecido pelo sistema operacional. O sistema operacional sabe o endereço IP do computador no qual o processo-cliente está sendo executado. O número de porta, no entanto, é um número inteiro de 16 *bits* temporário atribuído a um processo-cliente cada vez que o processo necessita iniciar a comunicação. O número da porta precisa ser atribuído a partir de um conjunto de números inteiros definidos pelas autoridades da Internet e chamados números efêmeros (temporários) de porta, que discutiremos mais adiante no Capítulo 3. O sistema operacional, entretanto, precisa garantir que o novo número de porta não seja usado por qualquer outro processo-cliente já em execução.

Endereço de *socket* remoto Encontrar o endereço de *socket* remoto (servidor) para um cliente, no entanto, exige mais trabalho. Quando um processo-cliente é iniciado, ele deve saber o endereço do *socket* servidor ao qual quer se conectar. Temos duas situações nesse caso.

- Em algumas ocasiões, o usuário que inicia o processo-cliente sabe o número de porta do servidor e o endereço IP do computador no qual o servidor está sendo executado. Isto geralmente ocorre em situações em que escrevemos o código das aplicações cliente e servidor e queremos testá-las. Por exemplo, no final deste capítulo, escrevemos alguns programas cliente e servidor simples e os testamos usando esse método. Nessa situação, o programador pode fornecer essas informações quando o programa-cliente é executado.
- Embora cada tipo de aplicação-padrão tenha um número de porta bem conhecido, na maioria das vezes não sabemos o endereço IP; isto acontece quando precisamos acessar uma página na Web, enviar um e-mail a um amigo, copiar um arquivo de um *site* remoto, por exemplo. Nessas situações, o servidor tem um nome, um identificador que define exclusivamente o processo-servidor, por exemplo, os URLs, como www.xxx.yyy, ou endereços de e-mail, como xxxx@yyyy.com. O processo-cliente precisa, portanto, alterar o identificador (nome) para o endereço de *socket* correspondente no servidor. Normalmente, o processo--cliente sabe o número da porta porque ele deve ser bastante conhecido, enquanto o endereço IP pode ser obtido usando outro aplicativo cliente-servidor chamado Sistema de Nomes de Domínio (DNS – Domain Name System). Discutiremos o DNS ainda neste capítulo, mas por enquanto é suficiente saber que ele atua como uma lista de nomes na Internet. Compare essa situação com a lista telefônica. Queremos ligar para alguém cujo nome sabemos, e cujo número de telefone pode ser obtido a partir da lista telefônica, que mapeia o nome para o número de telefone; o DNS mapeia o nome do servidor para o endereço IP do computador no qual o servidor está sendo executado.

2.2.2 Usando serviços da camada de transporte

Um par de processos fornece serviços aos usuários da Internet, sejam eles humanos ou programas. Mas ele precisa utilizar os serviços fornecidos pela camada de transporte para a comunicação, pois não há comunicação física na camada de aplicação. Como discutimos brevemente no Capítulo 1 e ainda discutiremos em detalhes no Capítulo 3, existem três protocolos de transporte comuns na pilha TCP/IP: UDP, TCP e SCTP. A maioria dos aplicativos padronizados foi projetada para utilizar os serviços de um desses protocolos. Quando escrevemos uma nova aplicação, podemos decidir qual protocolo queremos usar; a escolha do protocolo de camada de transporte afeta seriamente as características dos processos de aplicação. Nesta seção, discutimos primeiro os serviços fornecidos por cada protocolo para ajudar a entender por que um aplicativo padronizado os utiliza ou qual deles precisamos usar se decidirmos escrever um novo.

Protocolo UDP

O UDP fornece um serviço de datagrama não orientado à conexão e não confiável. Serviço não orientado à conexão significa que não existe qualquer conexão lógica entre os dois sistemas finais trocando mensagens. Cada mensagem é uma entidade independente encapsulada em um pacote denominado **datagrama**. O UDP não enxerga qualquer relação (ligação) entre os datagramas consecutivos vindos de uma mesma fonte e indo para o mesmo destino.

O UDP não é um protocolo confiável. Embora ele possa verificar se os dados não foram corrompidos durante a transmissão, ele não pede ao remetente para reenviar datagramas corrompidos ou perdidos. Para algumas aplicações, o UDP tem uma vantagem: ele é orientado à mensagem. Ele delimita a fronteira das mensagens trocadas.

Podemos comparar esse serviço não orientado à conexão e não confiável com o serviço normal fornecido pelos Correios. Duas entidades podem trocar várias cartas entre si, mas os Correios não veem qualquer conexão entre estas cartas. Para os Correios, cada carta é uma entidade separada com seu próprio remetente e destinatário. Se uma carta for perdida ou danificada durante a entrega, os correios não são responsáveis, apesar de tentaram dar o melhor de si durante a entrega.

Uma aplicação pode ser projetada para utilizar UDP se ela enviar mensagens pequenas e a simplicidade/velocidade forem mais importantes para a aplicação do que a confiabilidade. Por exemplo, algumas aplicações de gerenciamento e multimídia se encaixam nessa categoria.

Protocolo TCP

O TCP fornece um serviço de fluxo de *bytes* orientado à conexão e confiável. O TCP exige que dois sistemas finais primeiro criem uma conexão lógica entre eles por meio da troca de alguns pacotes de estabelecimento de conexão. Essa fase, geralmente denominada *handshaking* (que pode ser entendido como um "aperto de mãos"), estabelece alguns parâmetros entre os dois sistemas finais, incluindo o tamanho dos pacotes de dados a serem trocados, o tamanho dos *buffers* para armazenar os blocos de dados até a chegada da mensagem completa, e assim por diante. Após o processo de *handshaking*, as duas pontas podem enviar blocos de dados em ambas as direções na forma de segmentos. A continuidade dos *bytes* pode ser verificada pela numeração dos *bytes* trocados. Por exemplo, se alguns *bytes* forem perdidos ou danificados, o receptor pode solicitar o reenvio, o que faz com que o TCP seja um protocolo confiável. O TCP também pode fornecer controle de fluxo e controle de congestionamento, como veremos no Capítulo 3. Um problema com o protocolo TCP é que ele não é orientado a mensagens, ou seja, não delimita as fronteiras das mensagens trocadas.

Podemos comparar o serviço orientado à conexão e confiável fornecido pelo TCP com o serviço prestado pela companhia telefônica, embora apenas até certo ponto. Se duas partes decidem se comunicar por telefone em vez de usar os Correios, elas podem criar uma conexão e trocar palavras por um período de tempo. O serviço telefônico é, de certa forma, confiável, porque se uma pessoa não entende uma palavra ou algumas palavras pronunciadas pela outra, ela pode pedir a repetição.

A maioria das aplicações padronizadas que precisam enviar mensagens longas e exigem confiabilidade pode se beneficiar dos serviços do TCP.

Protocolo SCTP

O SCTP fornece um serviço que é uma combinação dos dois outros protocolos. Assim como no TCP, o serviço é orientado à conexão e confiável, mas não é orientado a fluxos de *bytes*. É um protocolo orientado a mensagens como o UDP. Além disso, o SCTP pode fornecer serviços de múltiplos fluxos por meio de múltiplas conexões da camada de rede.

Normalmente, o SCTP é adequado para qualquer aplicação que requer confiabilidade e, ao mesmo tempo, precisa manter uma conexão mesmo se houver uma falha em uma conexão da camada de rede.

2.3 APLICAÇÕES CLIENTE-SERVIDOR PADRONIZADAS

Desde a criação da Internet, vários programas cliente-servidor foram desenvolvidos. Não precisamos recriá-los, mas precisamos entender o que eles fazem. Para cada aplicação, também precisamos saber as opções disponíveis. O estudo dessas aplicações e das formas como elas fornecem diferentes serviços pode ajudar a criar aplicações personalizadas no futuro.

Nesta seção, selecionamos seis programas da camada de aplicação padronizados. Começamos com o HTTP e a World Wide Web (WWW) porque eles são utilizados por quase todos os usuários da Internet. Apresentamos, então, a transferência de arquivos e aplicações de correio eletrônico, os quais têm elevada carga de tráfego na Internet. Em seguida, explicamos o *login* remoto e como ele pode ser realizado usando os protocolos TELNET e SSH. Finalmente, discutimos o DNS, que é utilizado por todos os aplicativos para mapear o identificador da camada de aplicação para o endereço IP do *host* correspondente.

Algumas outras aplicações, como o Protocolo de Configuração Dinâmica de *Host* (DHCP – Dynamic Host Configuration Protocol) e o Protocolo Simples de Gerenciamento de Rede (SNMP – Simple Network Management Protocol), serão discutidas em outros capítulos.

2.3.1 World Wide Web e HTTP

Nesta seção, primeiramente apresentamos a World Wide Web (abreviada como WWW ou Web). Discutimos, então, o Protocolo de Transferência de Hipertexto (HTTP – HyperText Transfer Protocol), o programa cliente-servidor da camada de aplicação de modo geral usado no contexto da Web.

World Wide Web

A ideia da Web foi proposta pela primeira vez por Tim Berners-Lee, em 1989, no CERN*, a Organização Europeia para Pesquisa Nuclear, para permitir que vários pesquisadores em diferentes locais da Europa pudessem acessar as pesquisas uns dos outros. A Web comercial surgiu no início de 1990.

A Web é hoje um repositório de informações no qual os documentos, chamados de *páginas Web*, são distribuídos por todo o mundo e documentos relacionados são vinculados entre si. A popularidade e o crescimento da Web pode ser ligado a dois termos da frase anterior: *distribuídos* e *vinculados*. A distribuição permite o crescimento da Web. Cada servidor Web no mundo pode adicionar uma nova página Web ao repositório e anunciá-lo a todos os usuários da Internet sem sobrecarregar alguns poucos servidores. A vinculação permite que uma página Web faça referência a outra armazenada em um servidor em algum outro lugar no mundo. A vinculação de páginas Web é obtida usando um conceito chamado *hipertexto*, que foi introduzido muitos anos antes do advento da Internet. A ideia era utilizar uma máquina que automaticamente recuperasse outro documento armazenado no sistema quando um vínculo (*link*) aparecesse no documento. A Web implementou a ideia eletronicamente, permitindo que o documento vinculado fosse recuperado quando o *link* fosse clicado pelo usuário. Hoje, o termo *hipertexto*, cunhado para designar documentos de texto vinculados, foi alterado para *hipermídia*, para mostrar que uma página Web pode ser um documento de texto, uma imagem, um arquivo de áudio ou um arquivo de vídeo.

O objetivo da Web extrapolou a simples recuperação de documentos vinculados. Hoje, a Web é usada para prover serviços de compras eletrônicas e jogos; nela, é possível ouvir programas de rádio ou ver programas de televisão sempre que se desejar, sem ser obrigado a ouvir ou ver esses programas quando eles são originalmente transmitidos.

Arquitetura

O WWW atual é um serviço cliente-servidor distribuído, em que um cliente usando um navegador (*browser*) pode acessar um serviço por meio de um servidor. No entanto, o serviço fornecido é distribuído entre muitos locais chamados *sites*. Cada *site* possui um ou mais documentos, chamados de **páginas Web**. Cada página Web, no entanto, pode conter algumas ligações ou *links* para outras páginas do mesmo ou de outros *sites*. Em outras palavras, uma página Web pode ser simples ou composta. A simples não tem *links* para outras páginas; a composta tem um ou mais *links* para outras páginas Web. Cada uma delas é um arquivo com um nome e um endereço.

Exemplo 2.2

Considere que precisamos buscar um documento científico que contém uma referência a outro arquivo de texto e uma referência a uma imagem grande. A Figura 2.8 ilustra essa situação.

O documento principal e a imagem são armazenados em dois arquivos separados no mesmo *site* (arquivo A e arquivo B); o arquivo de texto referenciado é armazenado em outro *site* (arquivo C). Como estamos lidando com três arquivos diferentes, precisamos de três transações se quisermos ver o documento completo. A primeira

* Em francês: Conseil Européen pour la Recherche Nucléaire.

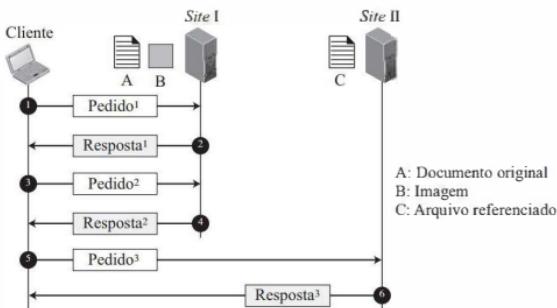



Figura 2.8 Esquema para o Exemplo 2.2.

transação (pedido/ resposta) recupera uma cópia do documento principal (arquivo A), que tem referências (ponteiros) para o segundo e terceiro arquivos. Após recuperar uma cópia do documento principal e navegar por ela, o usuário pode clicar sobre a referência à imagem para invocar a segunda transação e recuperar uma cópia da imagem (arquivo B). Se o usuário precisar ver o conteúdo do arquivo de texto referenciado, ele pode clicar sobre a sua referência (ponteiro) invocando a terceira operação e recuperando uma cópia do arquivo C. Observe que, apesar dos arquivos A e B serem armazenados no *site* I, eles são arquivos independentes com nomes e endereços distintos.

Duas transações são necessárias para recuperá-los. Um ponto muito importante de que precisamos lembrar é que o arquivo A, o arquivo B e o arquivo C no Exemplo 2.2 são páginas Web independentes, cada uma com um nome e endereço distintos. Embora as referências para os arquivos B ou C estejam inclusas no arquivo A, isto não significa que cada um desses arquivos não possa ser recuperado de forma independente. Um segundo usuário pode recuperar o arquivo B com uma transação. Um terceiro usuário pode recuperar o arquivo C com uma transação.

Cliente Web (Navegador) Diversos fornecedores oferecem **navegadores** (*browsers*) comerciais que interpretam e exibem uma página Web, e todos eles utilizam quase a mesma arquitetura. Cada navegador geralmente tem três partes: um controlador, protocolos cliente, e interpretadores. (Ver Figura 2.9.)

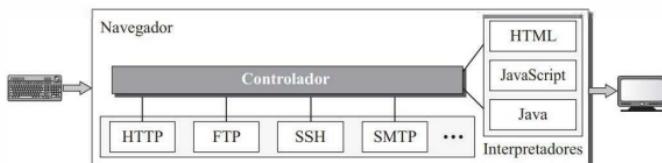



Figura 2.9 Navegador (*browser*).

O controlador recebe a entrada do teclado ou do mouse e usa os programas-cliente para acessar o documento. Depois, o controlador usa um dos interpretadores para exibir o documento na tela. O protocolo-cliente pode ser um dos protocolos descritos mais adiante, como HTTP ou FTP.

O interpretador pode ser HTML, Java, ou JavaScript, dependendo do tipo de documento. Alguns navegadores comerciais são o Internet Explorer, o Netscape Navigator, o Mozilla Firefox, o Google Chrome e o Opera.

Servidor Web A página Web é armazenada no servidor. Cada vez que um pedido chega, o documento correspondente é enviado ao cliente. Para conseguir uma melhor eficiência, os servidores normalmente armazenam arquivos solicitados em memória *cache* (memória de armazenamento temporário), pois é mais rápido acessar a memória do que o disco. Um servidor pode também se tornar mais eficiente se usar *multithreading* ou multiprocessamento. Nesse caso, um servidor pode responder a mais de um pedido de cada vez. Entre os servidores Web populares, estão o Apache e o Microsoft Internet Information Server (IIS).

Localizador Uniforme de Recursos (URL)

Uma página Web, assim como um arquivo, precisa ter um identificador único que permita distinguí-la de outras. Para definir uma página Web, precisamos de três identificadores: *host*, porta e caminho. Entretanto, antes da definição, precisamos dizer ao navegador que tipo de aplicação cliente-servidor queremos usar, o que é chamado de *protocolo*. Isto significa que precisamos de quatro identificadores: o primeiro é o tipo de instrumento que será utilizado para buscar a página Web; os três últimos formam uma combinação que define o objeto de destino (página Web).

- **Protocolo.** O primeiro identificador é a abreviação para o programa cliente-servidor que utilizamos para acessar a página Web. Embora na maior parte do tempo o protocolo seja o Protocolo de Transferência de Hipertexto (HTTP – HyperText Transfer Protocol), que discutiremos em breve, também podemos usar outros protocolos, como o Protocolo de Transferência de Arquivos (FTP – File Transfer Protocol).
- ***Host.*** O identificador do *host* pode ser o endereço IP do servidor ou o nome único dado ao servidor. Os endereços IP podem ser definidos usando notações decimais pontuadas, conforme descrito no Capítulo 4 (por exemplo, 64.23.56.17); o nome costuma ser o de domínio que define univocamente o *host*, tal como *forouzan.com*, algo que discutiremos quando falarmos do Sistema de Nomes de Domínio (DNS – Domain Name System) mais adiante.
- **Porta.** A porta, um número inteiro de 16 *bits*, normalmente é predefinida para a aplicação cliente-servidor. Por exemplo, se o protocolo HTTP é usado para acessar a página Web, o número de porta conhecido é 80. No entanto, se uma porta diferente for usada, o número pode ser fornecido explicitamente.
- **Caminho.** O caminho identifica o local e o nome do arquivo no sistema operacional subjacente. O formato do identificador normalmente depende do sistema operacional. No UNIX, um caminho é um conjunto de nomes de diretório seguido do nome do arquivo, todos separados por uma barra. Por exemplo, */topo/proximo/ultimo/meuarquivo* é um caminho que define univocamente um arquivo chamado *meuarquivo*, armazenado no diretório *ultimo*, que por sua vez é parte do diretório *proximo*, que por sua vez está dentro do diretório *topo*. Em outras palavras, o caminho lista os diretórios de cima para baixo, seguido então pelo nome do arquivo.

Para combinar essas quatro partes, foi criado o **Localizador Uniforme de Recursos** (URL – Uniform Resource Locator), que usa três diferentes separadores entre as quatro partes, conforme ilustrado a seguir:

protocolo://host/caminho	**Usado na maioria dos casos**
protocolo://host:**porta**/caminho	**Usado quando o número de porta é necessário**

Exemplo 2.3

O URL *http://www.mhhe.com/compsci/forouzan/* define a página Web relacionada a um dos autores deste livro. A cadeia de caracteres *www.mhhe.com* é o nome do computador na empresa McGraw-Hill (as três letras *www* são parte do nome do *host* e são adicionadas ao *host* comercial). O caminho é *compsci/forouzan/*, que define a página Web do Forouzan no diretório *compsci* (*computer science*, ou ciência da computação).

Documentos Web

Os documentos na WWW podem ser agrupados em três grandes categorias: estáticos, dinâmicos e ativos.

Documentos estáticos Documentos estáticos são documentos de conteúdo fixo criados e armazenados em um servidor. O cliente pode apenas obter uma cópia do documento. Em outras palavras, o conteúdo do arquivo é determinado quando este é criado, não quando é usado. Obviamente, os conteúdos no servidor podem ser alterados, mas o usuário não pode alterá-los. Quando um cliente acessa o documento, uma cópia dele é enviada. O usuário pode, então, usar um navegador para ver o documento. Documentos estáticos são preparados usando uma dentre várias linguagens: Linguagem de Marcação de Hipertexto (HTML – Hypertext Markup Language), Linguagem de Marcação Extensível (XML – Extensible Markup Language), Linguagem de Estilo Extensível (XSL – Extensible Style Language) e Linguagem de Marcação de Hipertexto Extensível (XHTML – Extensible Hypertext Markup Language). Discutiremos essas linguagens no Apêndice C, no *site* do livro.

Documentos dinâmicos Um documento dinâmico é criado por um servidor Web sempre que um navegador solicita o documento. Quando uma solicitação chega, o servidor Web executa um aplicativo ou um *script* que cria o documento dinamicamente. O servidor retorna o resultado do programa ou *script* como resposta para o navegador que solicitou o documento. Como um novo documento é criado para cada solicitação, o conteúdo de um documento dinâmico pode variar de um pedido para outro. Um exemplo muito simples de um documento dinâmico é uma página contendo a data e a hora de um servidor. A data e a hora são tipos de informações dinâmicas, dado que mudam a cada instante. O cliente pode pedir ao servidor que execute um programa tal como o **programa date** no UNIX, e então envie o resultado da execução de volta ao cliente. Embora a tecnologia de CGI (Common Gateway Interface) tenha sido bastante usada para recuperar documentos dinâmicos no passado, opções mais atuais incluem uma linguagem de *script*, tal como JSP (Java Server Pages), que usa a linguagem Java para executar *scripts*, ou ASP (Active Server Pages), um produto da Microsoft que utiliza a linguagem Visual Basic ou *ColdFusion*, que incorpora consultas a um banco de dados SQL (Structured Query Language, ou Linguagem de Consulta Estruturada) no documento HTML.

Documentos ativos Para muitas aplicações, precisamos que um programa ou um *script* seja executado no lado do cliente. Estes são chamados documentos ativos. Por exemplo, considere que queremos executar um programa que cria gráficos animados na tela ou um programa que interage com o usuário. O programa definitivamente precisa ser executado no lado do cliente, onde a animação ou interação ocorre. Quando um navegador solicita um documento ativo, o servidor envia uma cópia do documento ou um *script*, que é executado no lado do cliente (no navegador). Uma maneira de criar um documento ativo é utilizando *applets Java*, um programa escrito em Java no servidor. Ele já está compilado e pronto para ser executado. O documento encontra-se no formato de *bytecode* (binário). Outra forma é usando *JavaScripts*, mas, dessa vez, obtendo e executando o *script* no lado do cliente.

Protocolo de Transferência de Hipertexto

O **Protocolo de Transferência de Hipertexto** (HTTP – HyperText Transfer Protocol) é um protocolo usado para definir como os programas cliente-servidor podem ser escritos para recuperar

páginas da Web. Um cliente HTTP envia uma solicitação; um servidor HTTP retorna uma resposta. O servidor usa o número de porta 80; o cliente usa um número de porta temporário. O HTTP usa os serviços do TCP, o qual, conforme discutido anteriormente, é um protocolo orientado à conexão e confiável. Isto significa que, antes que ocorra qualquer transação entre o cliente e o servidor, uma conexão precisa ser estabelecida entre eles. Após a transação ser efetuada, a conexão deve ser encerrada, mas o cliente e o servidor não precisam se preocupar com erros nas mensagens trocadas ou com a perda de mensagens, já que o TCP é confiável e vai cuidar desses problemas, como veremos no Capítulo 3.

Conexões não persistentes *versus* persistentes

Conforme discutimos na seção anterior, o conceito de hipertexto incorporado em páginas Web pode exigir vários pedidos e respostas. Se as páginas Web, os objetos a serem recuperados, estiverem localizadas em servidores diferentes, não temos outra escolha além de criar uma nova conexão TCP para recuperar cada objeto. No entanto, se alguns dos objetos estão localizados no mesmo servidor, temos duas opções: recuperar cada objeto usando uma nova conexão TCP ou criar uma única conexão TCP e recuperá-los todos. O primeiro método é denominado *conexão não persistente*, e o segundo, *conexão persistente*. O HTTP especificava o uso de conexões *não persistentes* antes de sua versão 1.1, e as conexões *persistentes* são a opção-padrão da versão 1.1, apesar de isso poder ser alterado pelo usuário.

Conexões não persistentes Em uma conexão não persistente, uma conexão TCP é criada para cada pedido/resposta. Os passos desta estratégia são listados a seguir:

1. O cliente abre uma conexão TCP e envia uma solicitação.
2. O servidor envia a resposta e fecha a conexão.
3. O cliente lê os dados até encontrar um marcador de fim de arquivo; ele, então, fecha a conexão.

Nessa estratégia, se um arquivo contém *links* para N imagens diferentes em diferentes arquivos (todos localizados no mesmo servidor), a conexão deve ser aberta e fechada $N+1$ vezes. A estratégia não persistente impõe uma elevada carga computacional no servidor porque o servidor precisa de $N+1$ *buffers* diferentes cada vez que uma conexão é aberta.

Exemplo 2.4

A Figura 2.10 mostra um exemplo de uma conexão não persistente. O cliente precisa acessar um arquivo que contém um *link* para uma imagem. O arquivo de texto e a imagem estão localizados no mesmo servidor. Aqui, precisamos de duas conexões. O TCP exige pelo menos três mensagens de *handshake* para estabelecer a conexão, mas a solicitação pode ser enviada com a terceira mensagem. Depois que a conexão é estabelecida, o objeto pode ser transferido. Após receber um objeto, mais três mensagens de *handshake* são necessárias para encerrar a conexão, como veremos no Capítulo 3. Isto significa que o cliente e o servidor estão envolvidos em dois estabelecimentos de conexão e dois encerramentos de conexão. Se a transação envolve a recuperação de 10 ou 20 objetos, os tempos de ida e volta das mensagens envolvidas nesses *handshakes* se somam para criar uma grande carga computacional. Quando descrevermos a programação cliente-servidor no fim do capítulo, mostraremos que para cada conexão, o cliente e o servidor precisam alocar recursos adicionais, como *buffers* e variáveis. Essa é uma outra carga adicional em ambos os lados, mas especialmente no lado do servidor.

Conexões persistentes O HTTP versão 1.1 especifica como padrão o uso de uma conexão persistente. Em uma conexão persistente, o servidor deixa a conexão aberta para outras solicitações após o envio de uma resposta. O servidor pode fechar a conexão a pedido de um cliente ou se um tempo limite for atingido. O servidor geralmente envia o tamanho dos dados com cada resposta. No entanto, existem algumas ocasiões em que o servidor não conhece o tamanho dos dados, como quando

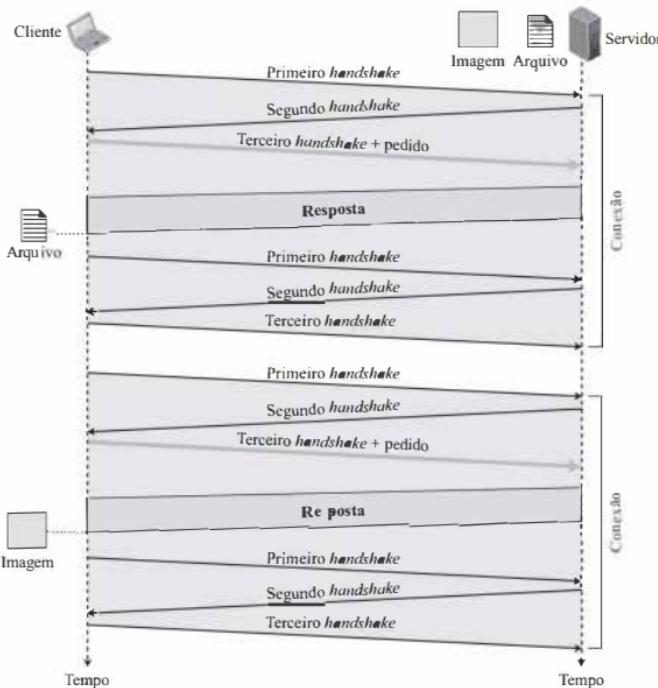



Figura 2.10 Esquema para o Exemplo 2.4.

um documento é criado dinâmica ou ativamente. Nesses casos, o servidor informa ao cliente que o tamanho não é conhecido e fecha a conexão após o envio dos dados, de modo que o cliente saiba que o final dos dados foi atingido. Tempo e recursos são economizados usando conexões persistentes. Apenas um conjunto de *buffers* e variáveis precisa ser definido para a conexão em cada local. O Tempo de Ida e Volta (RTT* – Round Trip Time) para o estabelecimento e para o encerramento da conexão é reduzido.

Exemplo 2.5

A Figura 2.11 mostra o mesmo cenário do Exemplo 2.4, mas usando uma conexão persistente. Apenas um estabelecimento e encerramento de conexão são usados, mas a solicitação da imagem é enviada separadamente.

* N. de T.: RTT corresponde ao tempo necessário para um pacote ir e voltar de seu destino. Esse conceito será mais bem explorado no Capítulo 3.

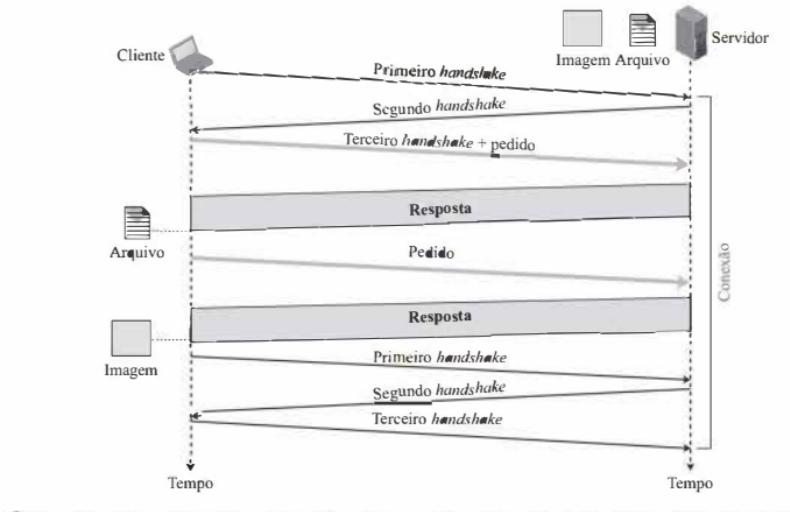



Figura 2.11 Esquema para o Exemplo 2.5.

Formatos de mensagens

O protocolo HTTP define o formato das mensagens de pedido e de resposta, conforme apresentado na Figura 2.12. Colocamos os dois formatos lado a lado para fins de comparação. Cada mensagem é composta de quatro seções: a primeira seção na mensagem de pedido é chamada de *linha de solicitação*, a primeira seção na mensagem de resposta é chamada de *linha de estado*. As outras três seções têm os mesmos nomes nas mensagens de pedido e de resposta. No entanto, as semelhanças entre elas estão apenas nos nomes, pois podem ter conteúdos diferentes. Discutiremos cada tipo de mensagem separadamente.

Mensagem de pedido Como dissemos anteriormente, a primeira linha em uma mensagem de pedido é chamada linha de solicitação. Existem três campos nessa linha separados por um espaço e terminados por dois caracteres – retorno de carro (*carriage return*) e nova linha (*line feed*) – como mostra a Figura 2.12. Os campos são chamados *método*, URL e *versão*.

O campo de método define os tipos de solicitação. Na versão 1.1 do HTTP, vários métodos são definidos, conforme mostra a Tabela 2.1. Na maioria das vezes, o cliente utiliza o método GET para enviar um pedido, e o corpo da mensagem fica vazio. O método HEAD é usado quando o cliente necessita apenas de algumas informações sobre a página Web do servidor, como a última vez em que ela foi modificada. Ele também pode ser usado para testar a validade de um URL. A mensagem de resposta, nesse caso, tem apenas a seção de cabeçalho, enquanto a seção de corpo fica vazia. O método PUT é o inverso do GET, permitindo ao cliente postar uma nova página Web no servidor (se for admitido). O método POST é semelhante ao PUT, mas é usado para enviar alguma informação para o servidor para ser adicionada à página Web ou para modificá-la. O método TRACE é utilizado para depuração; o cliente pede ao servidor que devolva o próprio pedido para verificar se o servidor está recebendo as solicitações. O método DELETE permite que o cliente remova uma página Web no servidor se o cliente tiver permissão para fazê-lo. O método CONNECT foi originalmente

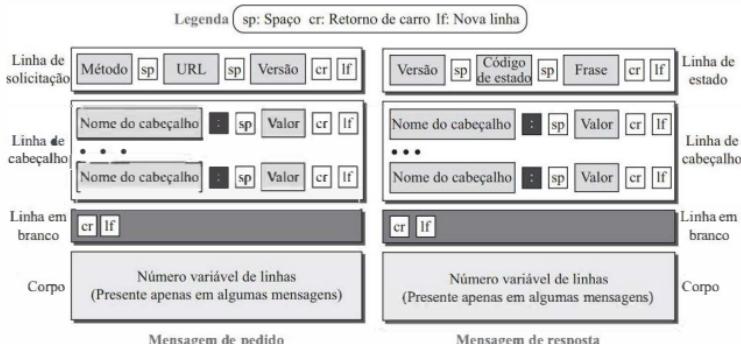



Figura 2.12 Formatos das mensagens de pedido e resposta.

criado como um método reservado; pode ser usado por servidores *proxy*, conforme discutido mais adiante. Finalmente, o método OPTIONS permite que o cliente pergunte sobre as propriedades de uma página Web.

Tabela 2.1 Métodos

Método	Ação
GET	Solicita um documento ao servidor
HEAD	Solicita informações sobre um documento, mas não o documento em si
PUT	Envia um documento do cliente para o servidor
POST	Envia alguma informação do cliente para o servidor
TRACE	Ecoa a solicitação recebida
DELETE	Remove a página Web
CONNECT	Reservado
OPTIONS	Consulta opções disponíveis

O segundo campo, URL, foi discutido anteriormente neste capítulo. Ele define o endereço e o nome da página Web correspondente. O terceiro campo, versão, mostra a versão do protocolo; a mais atual do HTTP é 1.1.

Após a linha de solicitação, podemos ter zero ou mais linhas de ***cabeçalho de solicitação***. Cada linha de cabeçalho envia informações adicionais do cliente para o servidor. Por exemplo, o cliente pode solicitar que o documento seja enviado em um formato especial. Cada linha de cabeçalho tem um nome de cabeçalho, um caractere de dois pontos (:), um espaço e um valor de cabeçalho (ver Figura 2.12). A Tabela 2.2 mostra alguns nomes de cabeçalho geralmente usados em uma solicitação. O campo valor define os valores associados a cada nome do cabeçalho. A lista de valores pode ser encontrada nas RFCs correspondentes.

O corpo pode estar presente em uma mensagem de solicitação, e costuma conter o comentário a ser enviado ou o arquivo a ser publicado no *site* quando o método é PUT ou POST.

Tabela 2.2 Nomes de cabeçalho de solicitação.

Cabeçalho	Descrição
User-agent	Identifica o programa-cliente
Accept	Mostra o formato de mídia que o cliente pode aceitar
Accept-charset	Mostra o conjunto de caracteres que o cliente pode manipular
Accept-encoding	Mostra o esquema de codificação que o cliente pode manipular
Accept-language	Mostra o idioma que o cliente pode aceitar
Authorization	Mostra quais permissões o cliente tem
Host	Mostra o *host* e o número de porta do cliente
Date	Mostra a data atual
Upgrade	Especifica o protocolo de comunicação preferencial
Cookie	Devolve o *cookie* para o servidor (explicado mais adiante)
If-Modified-Since	Se o arquivo foi modificado desde uma data específica

Mensagem de resposta O formato da mensagem de resposta também é mostrado na Figura 2.12. A mensagem de resposta consiste em uma linha de estado, linhas de cabeçalho, uma linha em branco, e às vezes um corpo. A primeira linha em uma mensagem de resposta é chamada *linha de estado*. Existem três campos nessa linha separados por espaços e terminados por um retorno de carro e uma nova de linha. O primeiro campo define a versão do protocolo HTTP, atualmente 1.1. O campo de código de estado define o estado do pedido. Ele consiste em três dígitos. Enquanto os códigos na faixa de 100 a 199 são apenas informativos, os códigos na faixa de 200 a 299 indicam o sucesso da solicitação. Os códigos na faixa de 300 a 399 redirecionam o cliente para outro URL, e os códigos na faixa de 400 a 499 indicam um erro no lado do cliente. Finalmente, os códigos na faixa de 500 a 599 indicam um erro no lado do servidor. A frase de estado explica o código do estado em formato de texto.

Após a linha de estado, pode haver zero ou mais linhas de *cabeçalho de resposta*, e cada uma envia informações adicionais do servidor para o cliente. Por exemplo, o remetente pode enviar informações adicionais sobre o documento. Cada linha de cabeçalho tem um nome de cabeçalho, um caractere de dois pontos (:), um espaço, e um valor do cabeçalho. Mostraremos algumas linhas de cabeçalho nos exemplos ao final desta seção. A Tabela 2.3 mostra alguns nomes de cabeçalho geralmente usados em uma mensagem de resposta.

Tabela 2.3 Nomes de cabeçalho de resposta.

Cabeçalho	Descrição
Date	Mostra a data atual
Upgrade	Especifica o protocolo de comunicação preferencial
Server	Fornece informações sobre o servidor
Set-Cookie	O servidor pede ao cliente que salve um *cookie*
Content-Encoding	Especifica o esquema de codificação

(*Continua*)

Tabela 2.3 Nomes de cabeçalho de resposta. *(Continuação)*	
Cabeçalho	**Descrição**
Content-Language	Especifica o idioma
Content-Length	Mostra o comprimento do documento
Content-Type	Especifica o tipo de mídia
Location	Para pedir ao cliente que envie o pedido a outro *site*
Accept-Ranges	O servidor aceitará as faixas de *byte* requisitadas
Last-modified	Fornece a data e a hora da última alteração

O corpo contém o documento a ser enviado pelo servidor para o cliente. O corpo está sempre presente, a não ser que a resposta seja uma mensagem de erro.

Exemplo 2.6

Esse exemplo recupera um documento (ver Figura 2.13). Usamos o método GET para recuperar uma imagem com o caminho */usr/bin/imagem1.* A linha de solicitação mostra o método (GET), o URL e a versão do HTTP (1.1). O cabeçalho tem duas linhas que mostram que o cliente pode aceitar imagens no formato GIF ou JPEG. O pedido não tem um corpo. A mensagem de resposta contém a linha de estado e quatro linhas de cabeçalho que definem a data, o servidor, a codificação do conteúdo (versão do MIME, que será descrito quando discutirmos sobre correio eletrônico) e o comprimento do documento. O corpo do documento vem depois do cabeçalho.

Figura 2.13 Esquema para o Exemplo 2.6.

Exemplo 2.7

Nesse exemplo, o cliente deseja enviar uma página Web para ser publicada no servidor. Usamos o método PUT, e a linha de solicitação mostra o método (PUT), o URL e a versão do HTTP (1.1). Existem quatro linhas de cabeçalhos. O corpo da solicitação contém a página Web a ser publicada. A mensagem de resposta contém a linha de estado e quatro linhas de cabeçalhos. O documento criado, que é um documento CGI, é fornecido como o corpo (ver Figura 2.14).

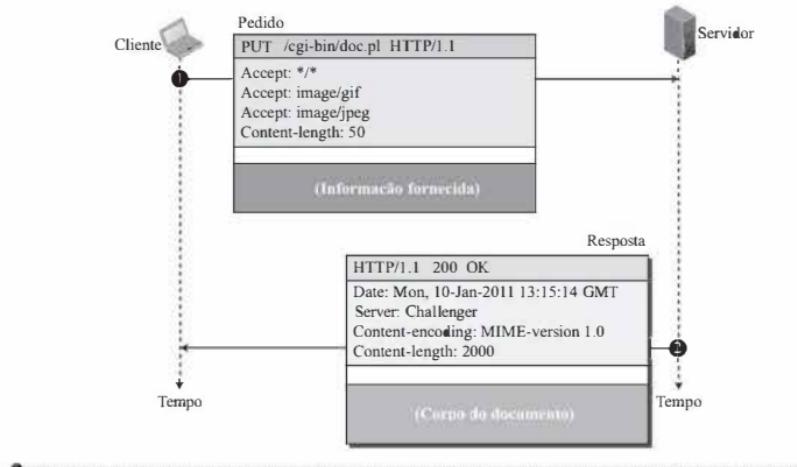



Figura 2.14 Esquema para o Exemplo 2.7.

Pedido condicional

Um cliente pode adicionar uma condição em seu pedido. Nesse caso, o servidor enviará a página Web solicitada se a condição for satisfeita, ou informará o cliente caso contrário. Uma das condições mais comuns impostas pelo cliente refere-se à data e hora em que a página Web foi modificada. O cliente pode enviar a linha de cabeçalho *If-Modified-Since* com o pedido para notificar ao servidor que precisa da página apenas se ela foi modificado após um certo ponto.

Exemplo 2.8

O exemplo a seguir mostra como um cliente pode impor a condição de data e hora de modificação em um pedido.

GET http://www.commonServer.com/information/arquivo1 HTTP/1.1	Linha de solicitação
If-Modified-Since: Thu, Sept 04 00:00:00 GMT	Linha de cabeçalho
	Linha em branco

A linha de estado na resposta mostra que o arquivo não foi modificado após o instante de tempo definido. O corpo da mensagem de resposta também se encontra vazio.

HTTP/1.1 304 Not Modified	Linha de estado
Date: Sat, Sept 06 08 16:22:46 GMT	Primeira linha de cabeçalho
Server: commonServer.com	Segunda linha de cabeçalho
	Linha em branco
(Corpo vazio)	Corpo vazio

Cookies

A World Wide Web foi originalmente concebida como uma entidade sem estado. Um cliente envia um pedido; um servidor responde, e o relacionamento entre as duas partes acaba aí. O propósito original da Web, a recuperação de documentos publicamente disponíveis, se encaixa perfeitamente nessa estrutura. Entretanto, a Web atual tem outras funcionalidades que precisam se lembrar de algumas informações sobre os clientes; algumas delas estão listadas a seguir:

- *Sites* estão sendo usados como *lojas virtuais*, permitindo que os usuários naveguem pela loja, selecionem os itens desejados, coloque-os em um carrinho de compras virtual e paguem no final com um cartão de crédito.
- Alguns *sites* precisam permitir o acesso apenas a *clientes registrados*.
- Alguns *sites* são usados como *portais*: o usuário seleciona as páginas Web que ele deseja ver.
- Alguns *sites* são apenas agências de *publicidade*.

Para esses fins, o mecanismo de *cookies* (cuja tradução literal seria "biscoito") foi criado.

Criando e armazenando *cookies* A criação e o armazenamento de *cookies* dependem da implementação; no entanto, o princípio básico permanece o mesmo.

1. Quando um servidor recebe um pedido de um cliente, ele armazena informações sobre o cliente em um arquivo ou em uma cadeia de caracteres. As informações podem incluir o nome de domínio do cliente, o conteúdo do *cookie* (informações que o servidor reuniu sobre o cliente, como nome, número da carteira de identidade etc.), um *timestamp* (registro de horário), e outras informações, dependendo da implementação.
2. O servidor inclui o *cookie* na resposta que envia ao cliente.
3. Quando o cliente recebe a resposta, seu navegador armazena o *cookie* no diretório de *cookies*, que é ordenado pelo nome de domínio do servidor.

Usando *cookies* Quando um cliente envia um pedido para um servidor, o navegador procura no diretório de *cookies* algum *cookie* enviado por aquele servidor. Caso seja encontrado, o *cookie* é incluído no pedido. Quando o servidor recebe o pedido, ele sabe que aquele é um cliente antigo, não um novo. Perceba que o conteúdo do *cookie* nunca é lido pelo navegador ou divulgado para o usuário. É um *cookie* "*preparado*" pelo servidor e "*comido*" pelo servidor. Agora, vejamos como um *cookie* é usado para os quatro propósitos mencionados anteriormente:

- Uma *loja virtual* (*e-commerce*) pode usar um *cookie* para seus clientes compradores. Quando um cliente escolhe um item e o insere em um carrinho, um *cookie* que contém informações sobre o item, como seu número e preço unitário, é enviado para o navegador. Se o cliente selecionar um segundo item, o *cookie* é atualizado com informações sobre a nova seleção, e assim por diante. Quando o cliente termina suas compras e deseja realizar o pagamento, o último *cookie* é recuperado e o preço total é calculado.
- O *site* que restringe o acesso apenas a *clientes cadastrados* envia o *cookie* para o cliente quando este se registra pela primeira vez. Para qualquer futuro acesso, apenas os clientes que enviam o *cookie* apropriado são autorizados.
- Um *portal* Web usa o *cookie* de uma maneira similar. Quando um usuário seleciona suas páginas favoritas, um *cookie* é criado e enviado. Se o *site* é acessado novamente, o *cookie* é enviado para o servidor para mostrar o que o cliente está procurando.
- *Cookies* também são usados por agências de *publicidade*. Uma agência de publicidade pode colocar anúncios (*banners*) publicitários em algum *site* principal que é frequentemente visitado por usuários. A agência de publicidade provê apenas um URL

que fornece o endereço da agência de publicidade e não o *banner* em si. Quando um usuário visita o *site* principal e clica no ícone de uma empresa, um pedido é enviado à agência de publicidade, que envia o *banner* solicitado, mas também inclui um *cookie* com o ID do usuário. Qualquer uso futuro dos *banners* adiciona informações ao banco de dados com relação ao perfil de comportamento daquele usuário na Web. A agência de publicidade pode compilar os interesses do usuário e então vender essa informação a terceiros. Esse uso de *cookies* os tornou bastante controverso. Espera-se que alguma nova regra seja elaborada para preservar a privacidade dos usuários.

Exemplo 2.9

A Figura 2.15 mostra um cenário em que uma loja virtual pode se beneficiar do uso de *cookies*.

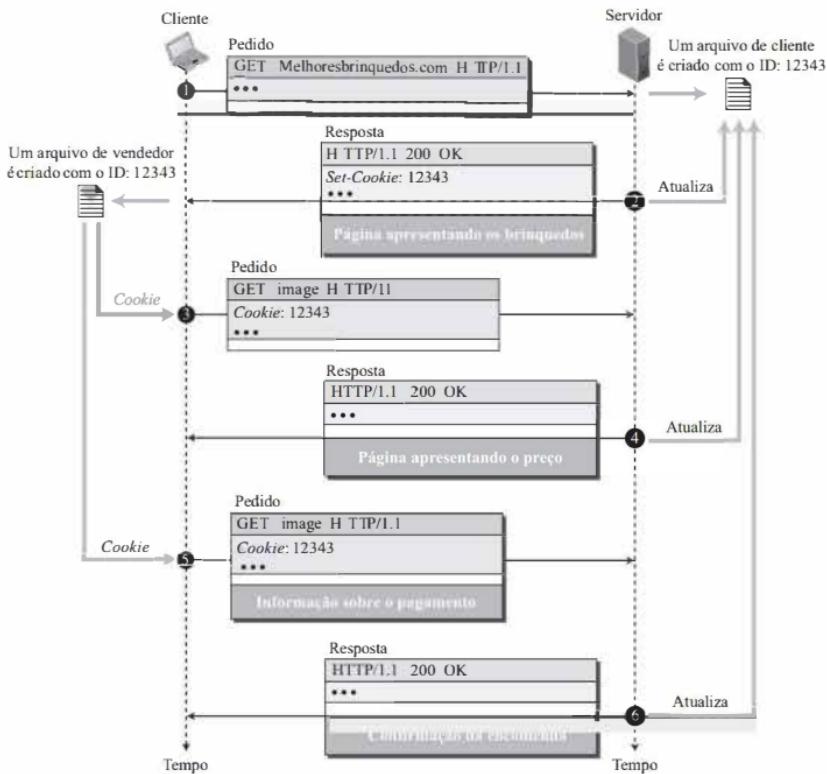



Figura 2.15 Esquema para o Exemplo 2.9.

Considere que um comprador quer adquirir um brinquedo de uma loja virtual chamada MelhoresBrinquedos. O navegador do comprador (cliente) envia um pedido para o servidor MelhoresBrinquedos. O servidor cria um carrinho de compras vazio (uma lista) para o cliente e atribui um ID para o carrinho de compras (por exemplo, 12343). O servidor, então, envia uma mensagem de resposta, que contém as imagens de todos os brinquedos disponíveis e um *link* associado a cada brinquedo; quando clicado, o *link* seleciona o brinquedo. A mensagem de resposta também inclui a linha de cabeçalho *Set-Cookie* cujo valor é 12343. O cliente exibe as imagens e armazena o valor do *cookie* em um arquivo chamado MelhoresBrinquedos. O *cookie* não é revelado ao comprador. Então, o comprador seleciona um dos brinquedos e clica sobre ele. O cliente envia um pedido, incluindo o ID 12343 na linha de cabeçalho de *Cookie*. Embora o servidor possa ter ficado ocupado e tenha esquecido desse comprador, ao receber o pedido e verificar o cabeçalho, ele encontra o valor 12343 como o valor do *cookie*. O servidor sabe que não se trata de um novo cliente; procura um carrinho de compras com o ID 12343. O carrinho de compras (lista) é aberto e o brinquedo selecionado é inserido na lista. O servidor envia, então, uma outra resposta ao comprador para informar o preço total e pedir a ele que efetue o pagamento. O comprador fornece informações sobre seu cartão de crédito e envia um novo pedido com o ID 12343 com o valor do *cookie*. Quando o pedido chega ao servidor, ele vê novamente o ID 12343 e aceita a encomenda e o pagamento, então envia uma confirmação na resposta. Outras informações sobre o cliente são armazenadas no servidor. Se o cliente acessar a loja no futuro, enviará o *cookie* novamente; a loja recuperará o arquivo e terá todas as informações sobre aquele cliente.

Armazenamento temporário na Web: servidor *proxy*

O HTTP suporta **servidores *proxy***. Um servidor *proxy* é um computador que mantém cópias de respostas a pedidos recentes. O cliente envia uma requisição HTTP ao servidor *proxy*, que verifica seu *cache* (memória de armazenamento temporário). Se a resposta não estiver armazenada no *cache*, o servidor *proxy* envia a solicitação para o servidor correspondente. Respostas recebidas são enviadas para o servidor *proxy* e armazenadas para futuras solicitações de outros clientes.

O servidor *proxy* reduz a carga no servidor original, diminui o tráfego e melhora a latência, mas, para usá-lo, o cliente deve ser configurado para acessar o *proxy* e não o servidor de destino.

Perceba que o servidor *proxy* funciona como servidor e cliente. Quando ele recebe de um cliente um pedido para o qual tem uma resposta, ele atua como um servidor e envia a resposta para o cliente. Quando recebe de um cliente um pedido para o qual ele não tem uma resposta, e primeiro atua como um cliente e envia um pedido para o servidor-alvo. Após a resposta ser recebida, ele atua novamente como um servidor e envia a resposta para o cliente.

Localização do servidor *proxy*

Os servidores *proxy* costumam ficar localizados no lado do cliente. Isso significa que podemos ter uma hierarquia de servidores *proxy*, como mostrado a seguir:

1. Um computador cliente pode também ser usado como um servidor *proxy*, com pequena capacidade, que armazena as respostas às solicitações frequentemente feitas pelo cliente.
2. Em uma empresa, um servidor *proxy* pode ser instalado na LAN para reduzir seu tráfego de entrada e saída.
3. Um ISP com muitos clientes pode instalar um servidor *proxy* para reduzir o tráfego de entrada e saída na rede daquele ISP.

Exemplo 2.10

A Figura 2.16 mostra um exemplo de uso de um servidor *proxy* em uma rede local, tal como uma rede em um *campus* ou em uma empresa. O servidor *proxy* está instalado na rede local. Quando uma solicitação HTTP é criada por qualquer um dos clientes (navegadores), o pedido é primeiramente direcionado para o servidor *proxy*;

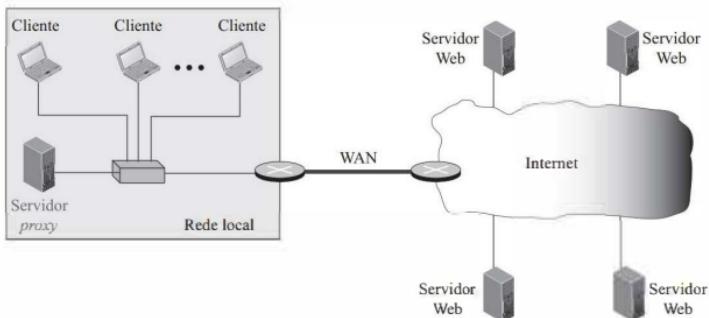



Figura 2.16 Exemplo de um servidor *proxy*.

se este já tem a página Web correspondente, envia a resposta ao cliente. Caso contrário, o servidor *proxy* funciona como um cliente e envia o pedido ao servidor Web na Internet. Quando a resposta é devolvida, o servidor *proxy* cria uma cópia dela e a armazena em seu *cache* antes de enviá-la para o cliente solicitante.

Atualização de *cache*

Uma questão muito importante é quanto tempo uma resposta deve permanecer no servidor *proxy* antes de ser eliminada e substituída. Várias estratégias diferentes são usadas para essa finalidade. Uma solução é armazenar a lista de *sites* cuja informação permanece inalterada por algum tempo. Por exemplo, uma agência de notícias pode mudar sua página de notícias todas as manhãs. Isso significa que um servidor *proxy* pode receber as notícias no início da manhã e mantê-las até o dia seguinte. Outra recomendação é adicionar alguns cabeçalhos para mostrar o instante da última modificação das informações. O servidor *proxy* pode, então, usar as informações contidas no cabeçalho para estimar por quanto tempo a informação pode ser válida.

Segurança do HTTP

O HTTP em si não fornece segurança. Entretanto, como mostraremos no Capítulo 10, o HTTP pode ser executado sobre a Camada de *Sockets* Segura (SSL – Secure Layer). Nesse caso, o HTTP é denominado HTTPS, que fornece confidencialidade, autenticação do cliente/servidor e integridade dos dados.

2.3.2 FTP

O **Protocolo de Transferência de Arquivos** (FTP – File Transfer Protocol) é o protocolo-padrão fornecido pelo TCP/IP para copiar um arquivo de uma estação para outra. Embora a transferência de arquivos de um sistema para outro possa parecer simples e direta, alguns problemas devem ser tratados primeiramente. Por exemplo, dois sistemas podem usar convenções diferentes para os nomes de arquivos. Dois sistemas podem ter diferentes maneiras de representar dados. Dois sistemas podem ter diferentes estruturas de diretório. Todos esses problemas foram resolvidos pelo FTP usando uma abordagem muito simples e elegante. Embora possamos transferir arquivos usando HTTP, o FTP é uma escolha melhor para transferir arquivos grandes ou que usam formatos diferentes. A Figura 2.17 mostra o modelo básico do FTP. O cliente tem três componentes: interface

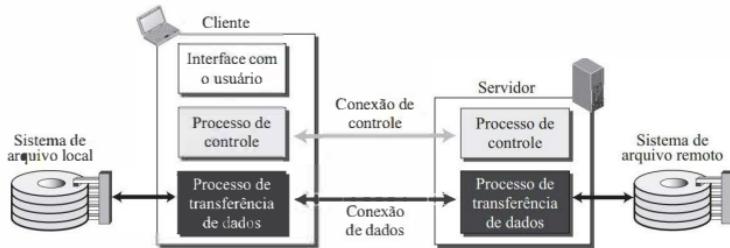



Figura 2.17 FTP

com o usuário, processo de controle cliente e processo de transferência de dados cliente. O servidor tem dois componentes: processo de controle servidor e processo de transferência de dados servidor. A conexão de controle é feita entre os processos de controle. A conexão de dados é feita entre os processos de transferência de dados.

A separação entre comandos e transferência de dados faz com que o FTP seja mais eficiente. A conexão de controle utiliza regras de comunicação muito simples. Precisamos transferir apenas uma linha de comando ou uma linha de resposta de cada vez. A conexão de dados, por outro lado, necessita de regras mais complexas devido à grande variedade de tipos de dados transferidos.

Tempos de vida das duas conexões

As duas conexões no FTP têm tempos de vida diferentes. A conexão de controle permanece ativa durante toda a interação da sessão FTP. A conexão de dados é aberta e em seguida fechada para cada atividade de transferência de arquivos. Ela é aberta toda vez que algum comando que envolve a transferência de arquivos é usado, e fechada quando o arquivo termina de ser transferido. Em outras palavras, quando um usuário inicia uma sessão FTP, a conexão de controle é aberta. Enquanto a conexão de controle estiver aberta, a conexão de dados pode ser aberta e fechada várias vezes se vários arquivos forem transferidos. O FTP usa duas portas TCP bem conhecidas: a porta 21 é usada para a conexão de controle, e a porta 20 é usada para a conexão de dados.

Conexão de controle

Para a comunicação de controle, o FTP usa a mesma abordagem que o TELNET (discutido mais adiante). Ele utiliza o conjunto de caracteres NVT ASCII, assim como o TELNET. A comunicação é obtida por meio de comandos e respostas. Esse método simples é adequado para a conexão de controle porque enviamos um comando (ou resposta) de cada vez. Cada linha é terminada com um indicador de fim de linha formado por dois caracteres (retorno de carro e nova linha).

Durante essa conexão de controle, os comandos são enviados do cliente para o servidor e as respostas são enviadas do servidor para o cliente. Comandos, que são enviados a partir dos processos clientes FTP de controle, seguem o formato ASCII com letras maiúsculas, e podem ou não ser seguidos por um argumento. Alguns dos comandos mais comuns são mostrados na Tabela 2.4.

Todo comando FTP gera pelo menos uma resposta. A resposta tem duas partes: um número de três dígitos seguido por texto. A parte numérica define o código; a parte textual define os parâmetros necessários ou explicações adicionais. O primeiro dígito define o estado do comando; o segundo dígito define a área em que o estado se aplica; e o terceiro dígito fornece informações adicionais. A Tabela 2.5 mostra algumas respostas comuns.

Tabela 2.4 Alguns coma

Comando	Argumento(s)	
ABOR		Aborta o comando anterior
CDUP		Alterna para o diretório pai
CWD	Nome do diretório	Alterna para outro diretório
DELE	Nome do arquivo	Exclui um arquivo
LIST	Nome do diretório	Lista subdiretórios ou arquivos
MKD	Nome do diretório	Cria um novo diretório
PASS	Usuário de Senha	Senha
PASV		Servidor escolhe uma porta
PORT	Identificador de porta	Cliente escolhe uma porta
PWD		Mostra o nome do diretório atual
QUIT		Sai do sistema
RETR	Nome(s) do(s) arquivo(s)	Recupera arquivos; os arquivos são transferidos do servidor para o cliente
RMD	Nome do diretório	Exclui um diretório
RNFR	Nome do arquivo (antigo)	Identifica um arquivo a ser renomeado
RNTO	Nome do arquivo (novo)	Renomeia o arquivo
STOR	Nome(s) do(s) arquivo(s)	Armazena arquivos; os arquivos são transferidos do cliente para o servidor
STRU	**F**, **R**, ou **P**	Define organização dos dados (**F**: arquivo, **R**: registro, ou **P**: página)
TYPE	**A**, **E**, **I**	Tipo padrão de arquivo (**A**: ASCII, **E**: EBCDIC, **I**: imagem)
USER	ID de usuário	Informações do usuário
MODE	**S**, **B**, ou **C**	Define modo de transmissão (**S**: fluxo, **B**: bloco, ou **C**: comprimido)

Tabela 2.5 Algumas respostas no FTP

Código	Descrição	Código	Descrição
125	Conexão de dados aberta	**250**	Solicitação de ação em arquivo OK
150	Estado do arquivo OK	**331**	Nome de usuário OK; senha necessária
200	Comando OK	**425**	Conexão de dados não pode ser aberta

(*Continua*)

Tabela 2.5 Algumas respostas no FTP (*Continuação*)

Código	Descrição	Código	Descrição
220	Serviço pronto	**450**	Ação sobre arquivo não executada; arquivo não disponível
221	Serviço fechando	**452**	Ação abortada; espaço de armazenamento insuficiente
225	Conexão de dados aberta	**500**	Erro de sintaxe; comando não reconhecido
226	Fechando conexão de dados	**501**	Erro de sintaxe em parâmetros ou argumentos
230	Login de usuário OK	**530**	Usuário não está logado

Conexão de dados

A conexão de dados usa a porta bem conhecida 20 no lado do servidor. No entanto, a criação de uma conexão de dados é diferente da criação da conexão de controle. Mostramos os passos envolvidos a seguir:

1. O cliente, não o servidor, abre uma conexão passiva usando uma porta efêmera. Isso deve ser feito pelo cliente porque é ele quem envia os comandos para a transferência de arquivos.
2. O cliente envia seu número de porta para o servidor usando o comando PORT.
3. O servidor recebe o número da porta e abre uma conexão ativa usando a porta bem conhecida 20 e o número da porta efêmera recebido.

Comunicação por meio da conexão de dados

O objetivo e a implementação da conexão de dados são diferentes daqueles da conexão de controle. Queremos transferir arquivos por meio da conexão de dados. O cliente deve definir o tipo de arquivo a ser transferido, a estrutura dos dados e o modo de transmissão. Antes de enviar o arquivo por meio da conexão de dados, nós nos preparamos para a transmissão por meio da conexão de controle. O problema da heterogeneidade é resolvido definindo-se três atributos de comunicação: tipo de arquivo, estrutura de dados e modo de transmissão.

Estrutura de dados O FTP pode transferir um arquivo por meio da conexão de dados usando uma das seguintes interpretações da estrutura dos dados: *estrutura de arquivos*, *estrutura de registro* ou *estrutura de página*. O formato da estrutura de arquivo (usado por padrão) não tem qualquer estrutura. Trata-se de um fluxo contínuo de *bytes*. Na estrutura de registros, o arquivo é dividido em *registros*. Essa estrutura pode ser usada apenas com arquivos de texto. Na estrutura da página, o arquivo é dividido em páginas, cada uma com um número e um cabeçalho de página. As páginas podem ser armazenadas e acessadas aleatória ou sequencialmente.

Tipo de arquivo O FTP pode transferir um dos seguintes tipos de arquivos por meio da conexão de dados: arquivos ASCII, EBCDIC ou de imagem.

Modo de transmissão O FTP pode transferir um arquivo por meio da conexão de dados usando um dos seguintes três modos de transmissão: *modo de fluxo*, *de bloco* ou *comprimido*. O modo de fluxo é o padrão; os dados são entregues do FTP para o TCP como um fluxo contínuo de *bytes*. No modo de blocos, os dados podem ser entregues do FTP para o TCP em blocos. Neste caso, cada

bloco é precedido por um cabeçalho de 3 *bytes*. O primeiro *byte* é chamado *descritor de bloco*; os dois *bytes* seguintes definem o tamanho do bloco em *bytes*.

Transferência de arquivos

A transferência de arquivos ocorre por meio da conexão de dados sob o controle dos comandos enviados pela conexão de controle. No entanto, devemos lembrar que a transferência de arquivos em FTP significa uma dentre três coisas: *recuperação de um arquivo* (servidor para cliente), *armazenamento de um arquivo* (cliente para servidor) e *listagem de diretório* (servidor para cliente).

Exemplo 2.11

A Figura 2.18 mostra um exemplo de uso do FTP para recuperar um arquivo, com apenas um arquivo a ser transferido. A conexão de controle permanece aberta o tempo todo, mas a conexão de dados é aberta e fechada repetidamente. Consideramos que o arquivo é transferido em seis seções. Depois que todos os registros são transferidos, o processo de controle no servidor anuncia que a transferência do arquivo foi finalizada. Como o processo de controle no cliente não tem qualquer arquivo para recuperar, ele emite o comando QUIT, o qual faz com que a conexão do serviço seja fechada.

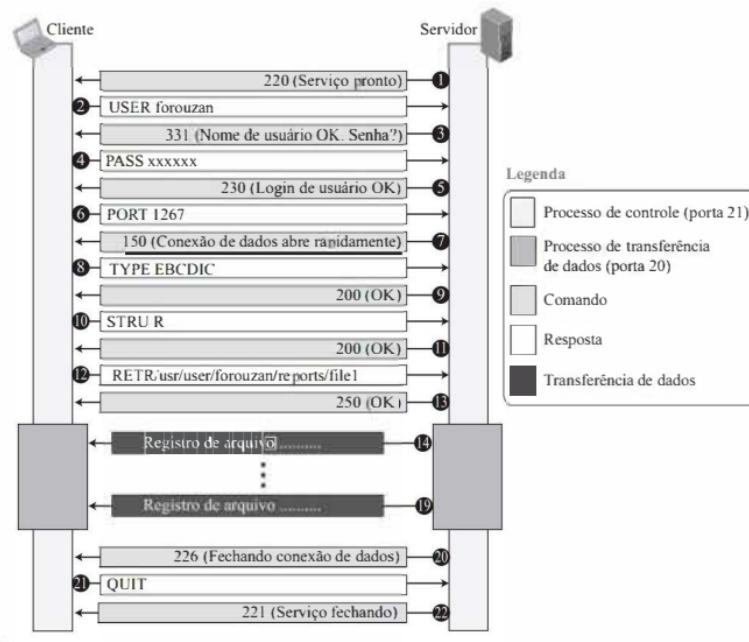



Figura 2.18 Esquema para o Exemplo 2.11.

Exemplo 2.12

A seguir, mostramos uma sessão FTP real na qual os diretórios são listados. As linhas cinzas mostram as respostas da conexão de controle do servidor; as linhas pretas, os comandos enviados pelo cliente; e as linhas em branco com fundo preto, a transferência de dados.

```
$ ftp voyager.deanza.fhda.edu
Conectado a voyager.deanza.fhda.edu.
220 (vsFTPd 1.2.1)
530 Por favor efetue o login usando USER e PASS.
Nome de usuário (voyager.deanza.fhda.edu:forouzan): forouzan
331 Por favor especifique a senha.
Senha:*********
230 Login realizado com sucesso.
Tipo do sistema remoto é UNIX.
Usando modo binário para transferir arquivos.
227 Entrando em Modo Passivo (153,18,17,11,238,169)
150 Aqui vai a lista de diretórios.
drwxr-xr-x  2  3027  411  4096  Sep 24  2002  negócios
drwxr-xr-x  2  3027  411  4096  Sep 24  2002  pessoal
drwxr-xr-x  2  3027  411  4096  Sep 24  2002  escola
226 Envio de diretório OK.
ftp> quit
221 Até logo.
```

Segurança no FTP

O protocolo FTP foi projetado quando a segurança não era um grande problema. Apesar de o FTP exigir uma senha, ela é enviada na forma de texto às claras (sem aplicação de criptografia), o que significa que pode ser interceptada e usada por um atacante. A conexão de transferência de dados também transfere dados em texto às claras, o que não é seguro. Para se tornar seguro, pode-se adicionar uma Camada de *Sockets* Segura (SSL – Secure Socket Layer) entre a camada de aplicação FTP e a camada TCP. Nesse caso, o FTP é chamado SSL-FTP. Também exploramos algumas aplicações de transferência segura de arquivos quando discutiremos o SSH mais adiante neste capítulo.

2.3.3 Correio eletrônico

O correio eletrônico (*electronic mail*, ou e-mail) permite que usuários troquem mensagens. A natureza dessa aplicação, contudo, é diferente das outras aplicações discutidas até agora. Em uma aplicação como o HTTP ou o FTP, o programa-servidor fica sendo executado o tempo todo, esperando por um pedido de um cliente. Quando o pedido chega, o servidor fornece o serviço. Existe um pedido e há uma resposta. No caso do correio eletrônico, a situação é diferente. Em primeiro lugar, o e-mail é considerado uma transação de mão única. Quando Alice envia um e-mail a Bob, ela pode até esperar uma resposta, mas esta não é obrigatória. Bob pode ou não responder; se responder, será outra transação de mão única. Em segundo lugar, não é viável nem lógico que Bob execute um

programa-servidor e espere até que alguém envie um e-mail para ele. Ele pode desligar seu computador quando não estiver em uso. Isto significa que a ideia de programação cliente-servidor deve ser implementada de outra forma: usando alguns computadores intermediários (servidores). Os usuários executam apenas programas-cliente quando eles quiserem e os servidores intermediários aplicam o paradigma cliente-servidor, como discutiremos na próxima seção.

Arquitetura

Para explicar a arquitetura de e-mail, vamos considerar um cenário comum, como o ilustrado na Figura 2.19. Outra possibilidade é o caso em que Alice (ou Bob) esteja diretamente conectada ao servidor de e-mail correspondente, caso em que a conexão LAN ou WAN não é obrigatória, mas essa variação no cenário não afeta nossa discussão.

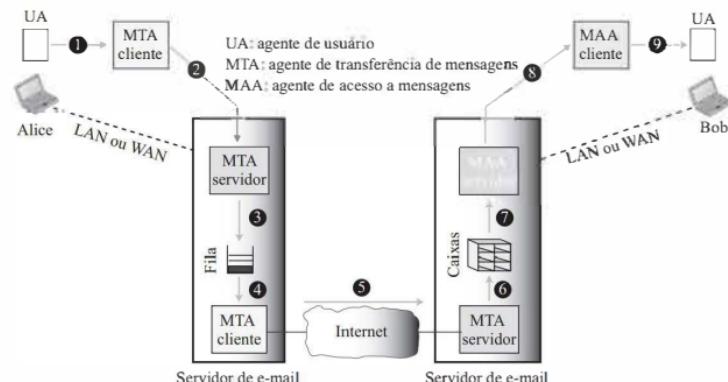



Figura 2.19 Cenário comum.

Nesse cenário comum, o remetente e destinatário do e-mail, Alice e Bob, respectivamente, são conectados por meio de uma LAN ou WAN a dois servidores de correio. O administrador criou uma caixa de correio para cada usuário, e nelas, as mensagens recebidas são armazenadas. Uma *caixa de correio* faz parte de uma unidade de disco rígido do servidor, um arquivo especial com restrições de acesso. Somente o dono da caixa de correio tem acesso a ela. O administrador também criou uma fila para armazenar mensagens a serem enviadas.

O envio de um simples e-mail de Alice para Bob requer nove diferentes passos, conforme mostrado na figura. Alice e Bob usam três diferentes *agentes*: um **Agente de Usuário** (UA – User Agent), um **Agente de Transferência de Mensagens** (MTA – Mail Transfer Agent), e um **Agente de Acesso a Mensagens** (MAA – Message Access Agent). Quando Alice precisa enviar uma mensagem a Bob, ela executa um programa AU para preparar a mensagem e enviá-la para seu servidor de e-mail. O servidor de e-mail no lado de Alice usa uma fila para armazenar as mensagens a serem enviadas. A mensagem, no entanto, precisa ser enviada via Internet, do lado de Alice para o lado de Bob, usando um MTA. Aqui, dois agentes de transferência de mensagens são necessários: um cliente e um servidor. Como a maioria dos programas cliente-servidor na Internet, o servidor precisa estar sendo executado o tempo todo, porque ele não sabe quando um cliente vai pedir uma conexão. O cliente, por outro lado, pode ser ativado pelo sistema quando houver uma mensagem na fila de envio. O agente de usuário no lado de Bob permite que ele leia a mensagem recebida.

Bob, posteriormente, usa um cliente MAA para recuperar a mensagem de um servidor MAA em execução no segundo servidor.

Há dois pontos importantes que precisamos enfatizar aqui. Em primeiro lugar, Bob não pode ignorar o servidor de e-mail e usar o servidor MTA diretamente, pois para isto, teria que executar o servidor MTA o tempo todo, por não saber quando uma mensagem vai chegar. Isto significa que Bob precisa manter seu computador ligado o tempo todo se ele estiver conectado ao seu sistema por meio de uma LAN. Se estiver conectado por meio de uma WAN, deve manter a conexão aberta o tempo todo. Nenhuma dessas situações é viável hoje.

Em segundo lugar, perceba que Bob precisa de outro par de programas cliente-servidor: programas de acesso à mensagem. Isto porque um programa cliente-servidor MTA é um programa do tipo *push*: o cliente "empurra" a mensagem para o servidor. Bob precisa de um programa do tipo *pull*. O cliente precisa "puxar" a mensagem do servidor. Discutiremos mais sobre MAAs em breve.

O sistema de correio eletrônico precisa de dois UAs, dois pares de MTAs (cliente e servidor), e um par de MAAs (cliente e servidor).

Agente de usuário

O primeiro componente de um sistema de correio eletrônico é o **agente de usuário** (UA – user agent). Ele presta serviços ao usuário para facilitar o processo de envio e recebimento de uma mensagem. Um agente é um pacote de *software* (um programa) que compõe, lê, responde, e encaminha mensagens. Ele também gerencia caixas de correio locais nos computadores dos usuários.

Existem dois tipos de agentes de usuário: aqueles com base em linha de comando e os com base em GUI (interface gráfica). Agentes de usuário com base em linha de comando apareceram nos primórdios do correio eletrônico, e ainda estão presentes como agentes de usuário subjacentes. Um agente de usuário com base em linha de comando normalmente aceita um comando de um caractere do teclado para executar sua tarefa. Por exemplo, um usuário pode digitar o caractere *r*, no prompt de comando, para responder ao remetente da mensagem, ou digitar o caractere *R* para responder ao remetente e a todos os destinatários. Alguns exemplos de agentes de usuário com base em linhas de comando são o *mail*, *pine*, e *elm*.

Agentes de usuário modernos têm base em Interface Gráfica de Usuário (GUI – Graphical User Interface). Eles contêm componentes de GUI que permitem ao usuário interagir com o *software* usando o teclado e o *mouse*. Eles apresentam componentes gráficos, como ícones, barras de menus e janelas que facilitam o acesso aos serviços. Alguns exemplos de agentes de usuário com base em GUI são *Mozilla Thunderbird*, *Eudora* e *Microsoft Outlook*.

Enviando e-mail

Para enviar um e-mail, o usuário, por meio do UA, cria uma entidade que se parece muito com uma carta comum. Ela contém um *envelope* e uma *mensagem* (ver Figura 2.20). O envelope geralmente contém o endereço do remetente, o endereço do destinatário e outras informações. A mensagem contém um *cabeçalho* e um *corpo*. O cabeçalho da mensagem define o remetente, o destinatário, o assunto da mensagem e algumas outras informações. O corpo da mensagem contém a informação de fato a ser lida pelo destinatário.

Recebendo e-mail

O agente de usuário é acionado pelo usuário (ou por um temporizador). Se um usuário tem um e-mail, o UA notifica o usuário. Se o usuário está pronto para ler o e-mail, uma lista é apresentada na qual cada linha contém um resumo das informações sobre uma determinada mensagem na caixa de correio. O resumo geralmente inclui o endereço de e-mail do remetente, o assunto e o instante

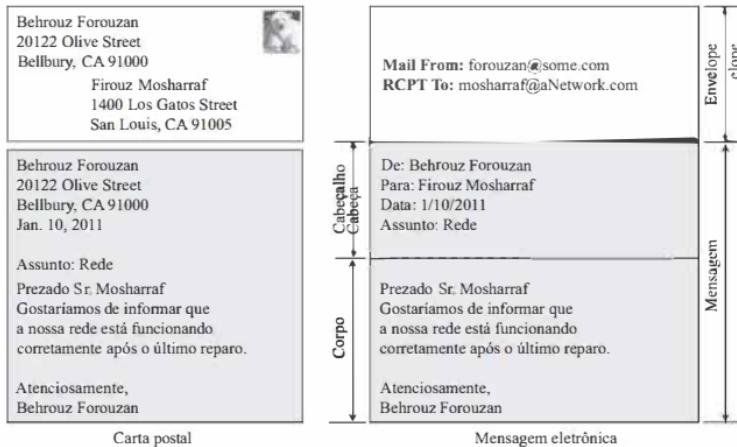



Figura 2.20 Formato de um e-mail.

em que o e-mail foi enviado ou recebido. O usuário pode selecionar qualquer uma das mensagens e exibir o seu conteúdo na tela.

Endereços

Para entregar o e-mail, um sistema de gerenciamento deve usar um sistema de endereçamento com endereços únicos. Na Internet, o endereço é composto por duas partes: uma *parte local* e um *nome de domínio*, separados por um sinal de @ (ver Figura 2.21).

Figura 2.21 Endereço de e-mail.

A parte local define o nome de um arquivo especial, chamado de caixa de correio do usuário, onde todo e-mail recebido por um usuário é armazenado para que possa ser recuperado pelo agente de acesso a mensagens. A segunda parte do endereço é o nome de domínio. Uma organização normalmente seleciona um ou mais *hosts* para receber e enviar e-mails; eles costumam ser chamados de servidores de e-mail. O nome de domínio atribuído a cada servidor de e-mail pode vir do banco de dados do DNS ou ser um nome lógico (por exemplo, o nome da organização).

Lista de discussão e lista de grupo

O serviço de correio eletrônico permite usar um *pseudônimo* para representar diversos endereços de e-mail; isto é chamado de lista de discussão. Cada vez que uma mensagem é enviada, o sistema

verifica o nome do destinatário considerando a base de dados de pseudônimos; se houver uma lista de discussão para o pseudônimo definido como destinatário, mensagens separadas, uma para cada entrada na lista, devem ser preparadas e entregues ao MTA.

Agente de transferência de mensagens: SMTP

Com base no cenário comum (Figura 2.19), podemos dizer que o e-mail é uma aplicação que precisa de três usos do paradigma cliente-servidor para realizar sua tarefa. É importante distinguir esses três usos quando estamos lidando com o e-mail. A Figura 2.22 mostra as três aplicações cliente-servidor correspondentes. Denominamos a primeira e a segunda como Agentes de Transferência de Mensagens (MTAs – Mail Transfer Agents), e a terceira como Agente de Acesso a Mensagens (MAA – Message Access Agent).

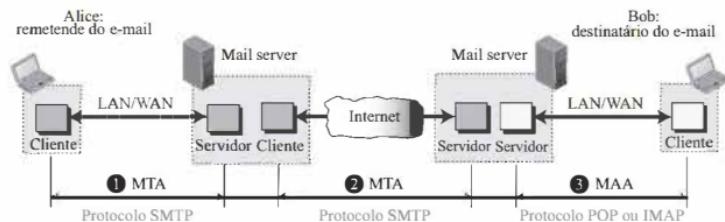



Figura 2.22 Protocolos usados no correio eletrônico.

O protocolo formal que define os MTAs cliente e servidor na Internet é chamado **Protocolo Simples de Transferência de Correio** (SMTP – Simple Mail Transfer Protocol). O SMTP é usado duas vezes, uma entre o remetente e o servidor de e-mail do remetente e outra entre os dois servidores de e-mail. Como veremos em breve, um outro protocolo é necessário entre o servidor de e-mail e o receptor. O SMTP simplesmente define como os comandos e as respostas devem ser enviados de um lado para outro.

Comandos e respostas

O SMTP usa comandos e respostas para transferir mensagens entre um MTA cliente e um MTA servidor. O comando vai de um MTA cliente para um MTA servidor; a resposta vai de um MTA servidor para o MTA cliente. Cada comando ou resposta é terminado por um indicador de fim de linha formado por dois caracteres (retorno de carro e nova linha).

Comandos Os comandos são enviados do cliente para o servidor. O formato de um comando é mostrado a seguir:

Palavra-chave: argumento(s)

Ele consiste em uma palavra-chave seguida de zero ou mais argumentos. O SMTP define 14 comandos, listados na Tabela 2.6.

Respostas As respostas são enviadas pelo servidor para o cliente. Uma resposta é um código de três dígitos que pode ser seguido por informação textual adicional. A Tabela 2.7 mostra os tipos de respostas mais comuns.

Tabela 2.6 Comandos SMTP.

Palavra-chave	Argumento(s)	Descrição
HELO	Nome do *host* do remetente	Identifica a si próprio
MAIL FROM	Remetente da mensagem	Identifica o remetente da mensagem
RCPT TO	Destinatário-alvo	Identifica o destinatário da mensagem
DATA	Corpo do e-mail	Envia a mensagem de fato
QUIT		Encerra a mensagem
RSET		Cancela a transação corrente de e-mail
VRFY	Nome do destinatário	Verifica o endereço do destinatário
NOOP		Verifica o estado do destinatário
TURN		Comuta o remetente e o destinatário
EXPN	Lista de discussão	Pede ao destinatário para expandir a lista de discussão
HELP	Nome do comando	Pede ao destinatário que envie informações sobre o comando enviado como argumento
SENO FROM	Destinatário-alvo	Especifica que o e-mail seja entregue apenas para o terminal do destinatário, e não para a caixa de correio
SMOL FROM	Destinatário-alvo	Especifica que o e-mail seja entregue ao terminal ou à caixa de correio do destinatário
SMAL FROM	Destinatário-alvo	Especifica que o e-mail seja entregue ao terminal e à caixa de correio do destinatário

Tabela 2.7 Respostas.

Código	Descrição
	Resposta de conclusão positiva
211	Estado do sistema ou resposta de ajuda
214	Mensagem de ajuda
220	Serviço pronto
221	Serviço fechando canal de transmissão
250	Comando solicitado completado
251	Usuário não local; a mensagem será encaminhada
	Resposta intermediária positiva
354	Iniciar entrada do e-mail
	Resposta de conclusão negativa temporária
421	Serviço não disponível

(*Continua*)

Tabela 2.7 Respostas. *(Continuação)*

Código	Descrição
450	Caixa de correio não disponível
451	Comando cancelado; erro local
452	Comando cancelado; espaço de armazenamento insuficiente
	Resposta de conclusão negativa permanente
500	Erro de sintaxe; comando não reconhecido
501	Erro de sintaxe em parâmetros ou argumentos
502	Comando não implementado
503	Sequência incorreta de comandos
504	Comando temporariamente não implementado
550	Comando não executado; caixa de correio indisponível
551	Usuário não local
552	Ação solicitada cancelada; espaço de armazenamento excedido
553	Ação solicitada não efetuada; nome da caixa de correio não permitido
554	Falha na transação

Fases da transferência de e-mail

O processo de transferência de uma mensagem de e-mail ocorre em três fases: estabelecimento de conexão, transferência de e-mail e encerramento de conexão.

Estabelecimento de conexão Depois que um cliente cria uma conexão TCP para a porta bem conhecida 25, o servidor SMTP inicia a fase de conexão. Esta fase envolve as seguintes três etapas:

1. O servidor envia o código 220 (serviço pronto) para dizer ao cliente que está pronto para receber mensagens de e-mail. Se o servidor não estiver pronto, ele envia o código 421 (serviço não disponível).
2. O cliente envia a mensagem HELO para se identificar, usando seu endereço de nome de domínio. Esse passo é necessário para informar o servidor do nome de domínio do cliente.
3. O servidor responde com o código 250 (comando solicitado completado) ou algum outro, dependendo da situação.

Transferência de mensagens Após a conexão ser estabelecida entre o cliente e o servidor SMTP, pode-se trocar uma única mensagem entre um remetente e um ou mais destinatários. Essa fase envolve oito passos. Os passos 3 e 4 são repetidos se houver mais do que um destinatário.

1. O cliente envia a mensagem MAIL FROM para especificar o remetente da mensagem. Ele inclui o endereço de e-mail do remetente (caixa de correio e nome de domínio). Esse passo é necessário para fornecer ao servidor o endereço de e-mail de resposta para retornar erros e notificações.
2. O servidor responde com o código 250 ou algum outro apropriado.

3. O cliente envia a mensagem RCPT TO (destinatário), a qual inclui o endereço de e-mail do destinatário.
4. O servidor responde com o código 250 ou algum outro apropriado.
5. O cliente envia a mensagem DATA para iniciar a transferência.
6. O servidor responde com o código 354 (iniciar entrada do e-mail) ou alguma outra mensagem apropriada.
7. O cliente envia o conteúdo da mensagem em linhas consecutivas. Cada linha termina com um indicador de fim de linha formado por dois caracteres (retorno de carro e nova linha). A mensagem termina com uma linha que contém apenas um ponto.
8. O servidor responde com o código 250 (OK) ou algum outro apropriado.

Encerramento da conexão Após a mensagem ser transferida com sucesso, o cliente finaliza a conexão. Essa fase envolve dois passos.

1. O cliente envia o comando QUIT.
2. O servidor responde com o código 221 ou algum outro apropriado.

Exemplo 2.13

Para mostrar as três fases da transferência de e-mail, exibimos todos os passos descritos anteriormente usando a informação ilustrada na Figura 2.23. Na figura, separamos as mensagens referentes ao envelope, cabeçalho e corpo na seção de transferência de dados. Observe que os passos nesta figura são repetidos duas vezes em cada transferência de e-mail: uma vez do remetente para o servidor de e-mail local e uma vez do servidor local para o servidor remoto. O servidor de e-mail local, após receber a mensagem de e-mail completa, pode colocá-la na fila e enviá-la para o servidor remoto em outro momento.

Agente de acesso a mensagens: POP e IMAP

O primeiro e o segundo estágios da entrega do e-mail usam o SMTP. No entanto, o SMTP não está envolvido na terceira fase, porque é um protocolo do tipo *push*; ele serve para enviar a mensagem do cliente para o servidor. Em outras palavras, a direção da maior parte dos dados (as mensagens) é do cliente para o servidor. Por outro lado, a terceira fase precisa de um protocolo do tipo *pull*; o cliente precisa recuperar as mensagens do servidor. A direção da maioria dos dados é do servidor para o cliente. A terceira etapa usa um agente de acesso a mensagens.

Atualmente, dois protocolos de acesso a e-mail estão disponíveis: o Protocolo de Agência de Correio, versão 3 (POP3 – Post Office Protocol) e o Protocolo de Acesso a Correio da Internet, versão 4 (IMAP4 – Internet Mail Access Protocol). A Figura 2.22 ilustra o papel desses dois protocolos.

POP3

O **Protocolo de Agência de Correio, versão 3** (POP3 – Post Office Protocol) é simples, porém limitado em termos de funcionalidade. O *software* cliente POP3 é instalado no computador do destinatário; o *software* servidor POP3 é instalado no servidor de e-mail.

O acesso ao e-mail começa com o cliente, quando o usuário precisa fazer o *download* de seu e-mail da caixa de correio presente no servidor. O cliente abre uma conexão com o servidor na porta TCP 110. Em seguida, envia seu nome de usuário e senha para acessar a caixa de correio. O usuário pode, então, listar e recuperar as mensagens de correio, uma a uma. A Figura 2.24 mostra um exemplo de *download* usando POP3. Ao contrário das outras figuras deste capítulo, colocamos o cliente no lado direito porque o destinatário do e-mail (Bob) está executando o processo-cliente para obter as mensagens do servidor remoto.

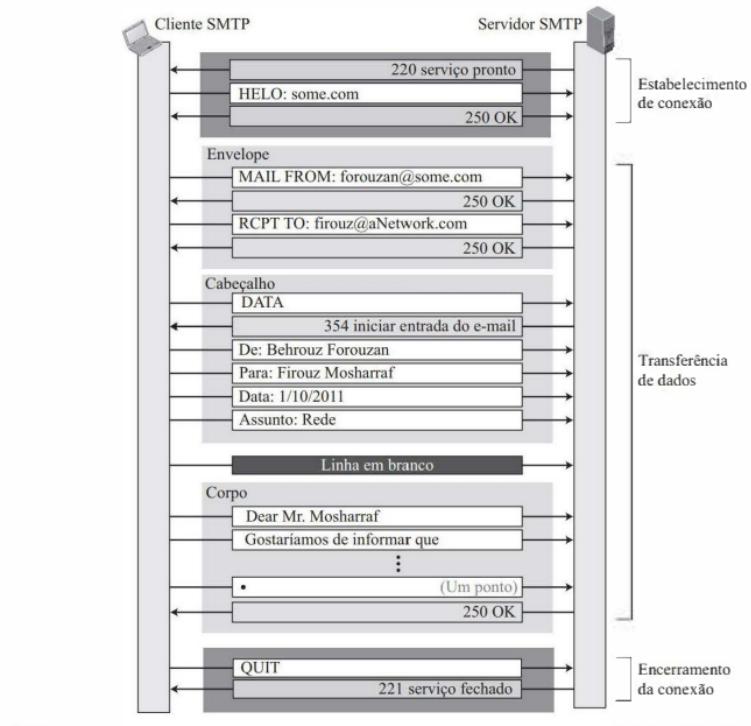



Figura 2.23 Esquema para o Exemplo 2.13.

O POP3 pode operar em dois modos: o modo *excluir* e o modo *manter*. No modo excluir, o e-mail é removido da caixa de correio após ser obtido. No modo manter, ele continua na caixa de correio após ser obtido. O modo excluir costuma ser utilizado quando o usuário está trabalhando em seu computador permanente e pode salvar e organizar os e-mails recebidos, depois de lê-los ou respondê-los. O modo manter é normalmente utilizado quando o usuário acessa a caixa de correio de um lugar diferente de seu computador principal (por exemplo, de um *laptop*). O e-mail é lido, mas mantido no sistema para ser obtido e organizado posteriormente.

IMAP4

Outro protocolo de acesso a e-mail é o **Protocolo de Acesso a Correio da Internet, versão 4** (IMAP4 – Internet Mail Access Protocol). O IMAP4 é semelhante ao POP3, mas tem mais funcionalidades. O IMAP4 é mais poderoso e mais complexo.

O POP3 apresenta várias deficiências. Ele não permite que o usuário organize seus e-mails no servidor, pois o usuário não pode ter diferentes pastas ali. Além disso, o POP3 não permite que o usuário verifique parcialmente o conteúdo do e-mail antes de obtê-lo. O IMAP4 fornece as seguintes funcionalidades extras:

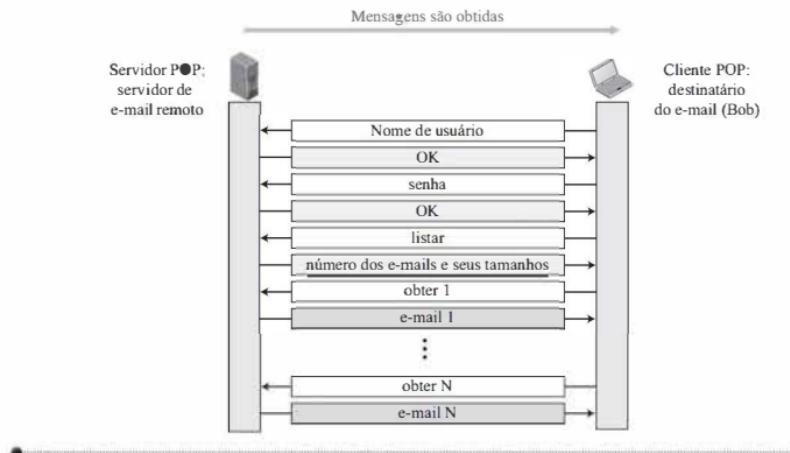



Figura 2.24 POP3

- Um usuário pode verificar o cabeçalho do e-mail antes de obtê-lo.
- Um usuário pode procurar por uma sequência específica de caracteres no conteúdo dos e-mails antes de obtê-los.
- Um usuário pode obter um e-mail parcialmente. Isto é especialmente útil se a largura de banda é limitada e o e-mail inclui conteúdo multimídia que exija um elevado consumo de banda.
- Um usuário pode criar, remover ou renomear caixas de correio no servidor de e-mail.
- Um usuário pode criar uma hierarquia de caixas de correio em uma pasta para armazenamento de e-mails.

MIME

O correio eletrônico tem uma estrutura simples, mas sua simplicidade tem um preço. Ele permite enviar apenas mensagens no formato NVT ASCII de 7 *bits*, ou seja, tem algumas limitações. Não pode ser usado para outros idiomas além do inglês (como francês, português, alemão, hebraico, russo, chinês e japonês). Além disso, ele não pode ser usado para enviar arquivos binários, de vídeo ou de áudio.

O **Extensões Multifunção para Mensagens de Internet** (MIME – Multipurpose Internet Mail Extensions) é um protocolo suplementar que permite que dados não ASCII sejam enviados por e-mail. O MIME transforma dados não ASCII do lado do remetente em dados NVT ASCII e entrega o resultado ao MTA cliente para ser enviado via Internet. A mensagem no lado do destinatário é transformada novamente, levando de volta aos dados originais.

Podemos pensar no MIME como um conjunto de funções de *software* que transformam dados não ASCII em dados ASCII e vice-versa, conforme ilustra a Figura 2.25.

Figura 2.25 MIME

Cabeçalhos MIME

O MIME define cinco cabeçalhos, mostrados na Figura 2.26, que podem ser adicionados à seção de cabeçalho do e-mail original para definir os parâmetros da transformação:

Figura 2.26 Cabeçalho **MIME**.

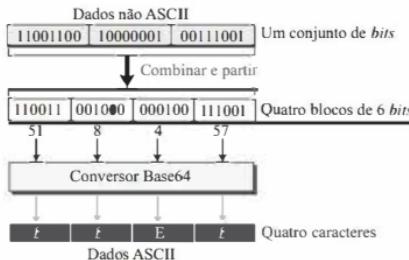
MIME-Version Este cabeçalho define a versão do MIME utilizado. A versão atual é a 1.1.

Content-Type Este cabeçalho define o tipo de dados usado no corpo da mensagem. O tipo de conteúdo e o subtipo de conteúdo são separados por uma barra (/). Dependendo do subtipo, o cabeçalho pode conter outros parâmetros. O MIME permite sete tipos diferentes de dados, listados na Tabela 2.8.

Tabela 2.8 Tipos e subtipos de dados no MIME.

Tipo	Subtipo	Descrição
Text	Plain	Texto não formatado
	HTML	Formato HTML (ver Apêndice C, no *site* do livro)
Multipart	Mixed	Corpo contém múltiplas partes ordenadas de diferentes tipos de dados
	Parallel	O mesmo que acima, mas sem ordenação das partes
	Digest	Similar ao Mixed, mas o tipo de dados-padrão torna-se message/rfc822
	Alternative	Indica que as múltiplas partes são diferentes versões do mesmo conteúdo
Message	RFC822	O corpo é uma mensagem encapsulada
	Partial	O corpo é um fragmento de uma mensagem maior
	External-Body	O corpo é uma referência a outra mensagem
Image	JPEG	Indica uma imagem no formato JPEG
	GIF	Indica uma imagem no formato GIF

(*Continua*)

Tabela 2.8 Tipos e subtipos de dados no MIME. *(Continuação)*

Tipo	Subtipo	Descrição
Video	MPEG	Indica um video no formato MPEG
Audio	Basic	Indica um dado de áudio com codificação em um único canal de voz de 8 KHz
Application	PostScript	Formato Adobe PostScript
	Octet-stream	Dados binários genéricos (*bytes* de oito *bits*)

Content-Transfer-Encoding Este cabeçalho define o método utilizado para codificar as mensagens em 0s e 1s para o transporte. Os cinco tipos de métodos de codificação estão listados na Tabela 2.9.

Tabela 2.9 Métodos de codificação do campo Content-Transfer-Encoding.

Tipo	Descrição
7-bit	Caracteres NVT ASCII com cada linha apresentando menos de 1.000 caracteres
8-bit	Caracteres não ASCII com cada linha apresentando menos de 1.000 caracteres
Binary	Caracteres não ASCII com linhas de tamanho ilimitado
Base64	Blocos de dados de 6 *bits* codificados em caracteres ASCII de 8 *bits*
Quoted-printable	Caracteres não ASCII codificados como um sinal de igual (=) acrescidos de um código ASCII

Os dois últimos métodos de codificação são interessantes. Na codificação Base64, dados, tais como uma sequência de *bits*, são primeiramente divididos em pedaços de 6 *bits*, conforme mostra a Figura 2.27.

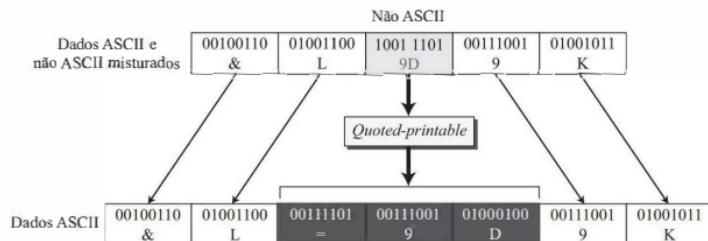



Figura 2.27 Conversão para Base64.

Cada pedaço de 6 *bits* é então convertido em um caractere ASCII de acordo com a Tabela 2.10.

O esquema de codificação Base64 apresenta redundâncias, ou seja, cada grupo de seis *bits* toma-se um caractere ASCII e é enviado como oito *bits*. Isto leva a uma sobrecarga ou *overhead* de 25%. Se os dados são formados principalmente por caracteres ASCII com uma pequena porção não ASCII, podemos usar a codificação *quoted-printable*. Com ela, se um caractere é ASCII, ele é enviado sem alteração. Se um caractere não é ASCII, ele é enviado na forma de três caracteres. O primeiro caractere é o sinal de igual (=). Os dois caracteres seguintes são a representação hexadecimal do *byte*. A Figura 2.28 mostra um exemplo. No exemplo, o terceiro caractere não é ASCII porque começa com o *bit* 1. Ele é interpretado como dois dígitos hexadecimais ($9D_{16}$), e é substituído por três caracteres ASCII (=, 9, D).

Tabela 2.10 Tabela de Conversão Base64.

Valor	Código	Valor	Código	Valor	Código	Valor	Código	Valor	Código	Valor	Código
0	A	11	L	22	W	33	h	44	s	55	3
1	B	12	M	23	X	34	i	45	t	56	4
2	C	13	N	24	Y	35	j	46	u	57	5
3	D	14	O	25	Z	36	k	47	v	58	6
4	E	15	P	26	a	37	l	48	w	59	7
5	F	16	Q	27	b	38	m	49	x	60	8
6	G	17	R	28	c	39	n	50	y	61	9
7	H	18	S	29	d	40	o	51	z	62	+
8	I	19	T	30	e	41	p	52	0	63	/
9	J	20	U	31	f	42	q	53	1		
10	K	21	V	32	g	43	r	54	2		

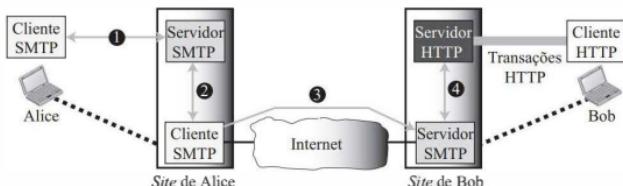
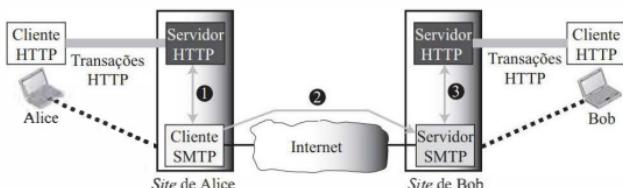



Figura 2.28 *Quoted-printable.*

Content-ID Este cabeçalho identifica univocamente a mensagem como um todo em um ambiente com várias mensagens.

Content-Description Este cabeçalho define se o corpo é uma imagem, um áudio ou um vídeo.

E-mail baseado na Web

O e-mail é uma aplicação tão comum que alguns *sites* na Web hoje oferecem esse serviço para qualquer um que o acesse. Três *sites* comuns são o Hotmail, Yahoo e o Google mail. A ideia é muito simples. A Figura 2.29 mostra dois casos:

Caso 1: Apenas o destinatário usa HTTP

Caso 2: Tanto remetente como destinatário usam HTTP

Figura 2.29 E-mail baseado na Web, casos I e II.

Caso I

No primeiro caso, Alice, a remetente, usa um servidor de e-mail tradicional; Bob, o destinatário, tem uma conta em um servidor de e-mail na Web. A transferência de e-mail do navegador de Alice para seu servidor é feita via SMTP. A transferência da mensagem do servidor emissor para o servidor receptor também é feita via SMTP. No entanto, a transferência da mensagem do servidor receptor (o servidor Web) para o navegador de Bob é feita via HTTP, ou seja, em vez de usar POP3 ou IMAP4, o HTTP é utilizado normalmente. Quando Bob precisa recuperar seus e-mails, ele envia uma mensagem de requisição HTTP para o *site* (Hotmail, por exemplo). O *site* envia um formulário que deve ser preenchido por Bob, incluindo seu nome de usuário e senha. Se os dados forem válidos, a lista de e-mails é transferida do servidor Web para o navegador de Bob no formato HTML. Agora, Bob pode navegar, visualizar seus e-mails recebidos e, em seguida, por meio de novas transações HTTP, pode obter seus e-mails um a um.

Caso II

No segundo caso, tanto Alice como Bob usam servidores Web, mas não necessariamente o mesmo servidor. Alice envia a mensagem para o servidor Web usando transações HTTP. Alice envia uma mensagem de requisição HTTP para seu servidor Web usando o nome e endereço da caixa de correio de Bob como o URL. O servidor no lado de Alice passa a mensagem para o cliente SMTP e a envia para o servidor no lado de Bob usando o protocolo SMTP. Bob recebe a mensagem usando transações HTTP. No entanto, a transferência da mensagem do servidor no lado de Alice para o servidor no lado de Bob ainda usa o protocolo SMTP.

Segurança de e-mail

Os protocolos discutidos neste capítulo não fornecem por si próprios quaisquer mecanismos de segurança. No entanto, as trocas de e-mail podem ser feitas de forma segura usando dois mecanismos

de segurança da camada de aplicação projetados especialmente para sistemas de e-mail. Dois protocolos, o Privacidade Muito Boa (PGP – Pretty Good Privacy) e o Extensões Seguras Multifunção para Mensagens de Internet (S/MIME – Secure/Multipurpose Internet Mail Extensions), serão discutidos no Capítulo 10, depois de abordarmos princípios básicos de segurança de redes.

2.3.4 TELNET

Um programa-servidor pode fornecer um serviço específico para o seu programa-cliente correspondente. Por exemplo, o servidor FTP é projetado para permitir que clientes FTP armazenem ou recuperem arquivos no lado do servidor. No entanto, é impossível ter um par cliente/servidor para cada tipo de serviço que necessitamos; o número de servidores se tornaria rapidamente intratável. A ideia não é escalável. Outra solução é ter um programa cliente/servidor específico para um conjunto de cenários comuns, mas ter alguns programas cliente/servidor genéricos que permitam que um usuário no lado do cliente acesse o computador no lado do servidor e utilize os serviços ali disponíveis. Por exemplo, se um estudante precisa utilizar o programa compilador de Java que está no laboratório da universidade, não há necessidade de um haver compilador Java cliente e um compilador Java servidor. O estudante pode usar um programa-cliente de acesso para entrar no servidor da universidade e usar o compilador como se estivesse na universidade. A esses pares cliente/servidor genéricos de aplicações damos o nome **acesso remoto**.

Um dos protocolos de acesso remoto originalmente criados é o **TELNET**, que é uma abreviatura para *TErminaL NETwork* (ou Terminal de Rede). Embora o TELNET exija um nome de usuário e senha, ele é vulnerável a ataques porque envia todos os dados, incluindo a senha, na forma de texto às claras (não criptografado). Um *hacker* pode espionar a rede e obter o nome de usuário e senha. Devido a esse problema de segurança, o uso de TELNET foi reduzido em favor de um outro protocolo, o SSH (Secure Shell), que descreveremos na próxima seção. Embora o TELNET esteja quase substituído pelo SSH, aqui discutiremos brevemente o TELNET por duas razões:

1. A arquitetura simples de texto às claras do TELNET permite explicar os problemas e desafios relacionados com o conceito de acesso remoto que também aparecem no SSH quando ele é usado como um protocolo de acesso remoto.
2. Administradores de rede costumam usar o TELNET para fins de diagnóstico e depuração.

Acesso local *versus* remoto

Primeiramente, discutimos o conceito de acesso local e remoto conforme ilustrado na Figura 2.30.

Quando um usuário faz *login* em um sistema local, isto é chamado de *acesso local*. À medida que um usuário digita em um terminal ou em uma estação de trabalho executando um emulador de terminal, as teclas pressionadas são aceitas pelo controlador do terminal, que passa os caracteres para o sistema operacional. Este, por sua vez, interpreta a combinação de caracteres e invoca o aplicativo ou utilitário desejado.

No entanto, quando um usuário quer acessar uma aplicação ou utilitário localizado em uma máquina remota, ela faz um *acesso remoto*. Neste cenário, os programas cliente e servidor TELNET entram em ação. O usuário envia as teclas pressionadas para o controlador do terminal onde o sistema operacional local aceita os caracteres, mas não os interpreta. Os caracteres são enviados para o cliente TELNET, que codifica os caracteres usando um conjunto de caracteres universal chamado Terminal Virtual de Rede (NVT – Network Virtual Terminal), discutido a seguir, e os entrega à pilha de protocolos TCP/IP local.

Os comandos ou texto, no formato NVT, viajam através da Internet e chegam à pilha TCP/IP da máquina remota. Lá, os caracteres são entregues ao sistema operacional e passados para o servidor TELNET, que traduz os caracteres para caracteres correspondentes compreensíveis pelo computador remoto. No entanto, os caracteres não podem ser passados diretamente para o sistema operacional

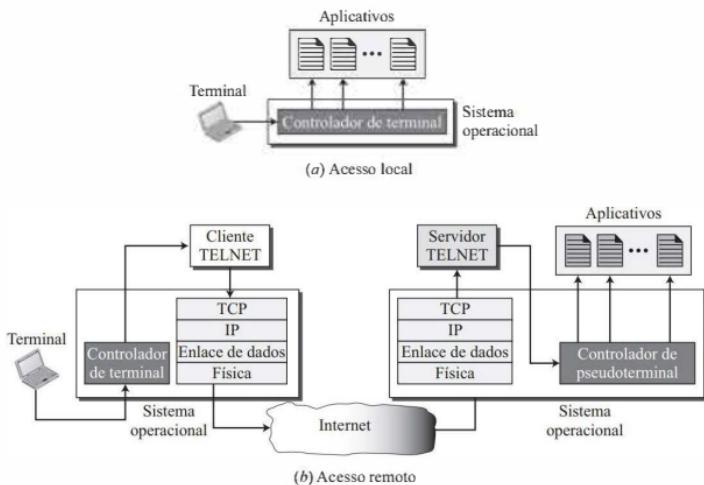



Figura 2.30 Acesso local *versus* remoto.

porque o sistema operacional remoto não foi projetado para receber caracteres de um servidor TELNET, mas, sim, para receber caracteres de um controlador de terminal. A solução é adicionar um *software* denominado *controlador de pseudoterminal*, que finge que os caracteres estão vindo de um terminal. O sistema operacional então passa os caracteres para o aplicativo apropriado.

Terminal Virtual de Rede

O mecanismo para acessar um computador remoto é complexo. Isso acontece porque cada computador e seu sistema operacional aceitam uma combinação especial de caracteres como palavras válidas, ou *símbolos*. Por exemplo, o símbolo de fim de arquivo em um computador executando o sistema operacional DOS é Ctrl+z, enquanto no sistema operacional UNIX, o mesmo símbolo é dado por Ctrl+d.

Estamos lidando com sistemas heterogêneos. Se quisermos acessar qualquer computador remoto no mundo, é preciso primeiro saber a qual tipo de computador vamos nos conectar, e também devemos instalar o emulador de terminal específico utilizado por aquele computador. O TELNET resolve esse problema pela definição de uma interface universal chamada conjunto de caracteres **Terminal Virtual de Rede** (NVT – Network Virtual Terminal). Por meio dela, o cliente TELNET traduz caracteres (dados ou comandos) que vêm do terminal local no formato NVT e os envia para a rede. O servidor TELNET, por outro lado, traduz os dados e comandos do formato NVT para o formato aceitável pelo computador remoto. A Figura 2.31 ilustra esse conceito.

O NVT utiliza dois conjuntos de caracteres, um para dados e outro para controle. Ambos são *bytes* de 8 *bits* conforme mostrado na Figura 2.31. Para dados, o NVT normalmente usa aquilo que é chamado NVT ASCII. Trata-se de um conjunto de caracteres de 8 *bits* no qual os sete *bits* menos significativos são os mesmos que os do US ASCII e o *bit* mais significativo é 0. Para enviar caracteres de controle entre computadores (do cliente para o servidor ou vice-versa), o NVT usa um conjunto de caracteres de 8 *bits* no qual o *bit* mais significativo é fixado em 1.

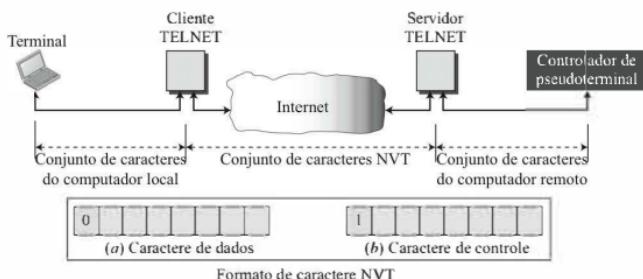



Figura 2.31 Conceito de NVT.

Opções

O TELNET permite que o cliente e o servidor negociem opções antes ou durante a utilização do serviço. As opções são recursos extras disponíveis para um usuário com um terminal mais sofisticado. Os usuários com terminais mais simples podem usar os recursos-padrão.

Interface do usuário

O sistema operacional (UNIX, por exemplo), define uma interface com comandos simples. Um exemplo de tal conjunto de comandos pode ser encontrado na Tabela 2.11.

Tabela 2.11 Exemplos de comandos de interface.

Comando	Significado	Comando	Significado
open	Conecta-se a um computador remoto	**set**	Define os parâmetros de operação
close	Fecha a conexão	**status**	Mostra informações de estado
display	Mostra os parâmetros operacionais	**send**	Envia caracteres especiais
mode	Alterna entre modo linha ou caractere	**quit**	Sai do TELNET

2.3.5 Secure Shell

Embora o SSH (Secure Shell) seja um programa de aplicação seguro que pode atualmente ser usado para diversas finalidades, tais como o acesso remoto e transferência de arquivos, ele foi originalmente concebido para substituir o TELNET. Existem duas versões do SSH: SSH-1 e SSH-2, totalmente incompatíveis. A primeira versão, o SSH-1, é agora obsoleta em função de suas falhas de segurança. Nesta seção, discutiremos apenas o SSH-2.

Componentes

O SSH é um protocolo de camada de aplicação com três componentes, conforme mostra a Figura 2.32.

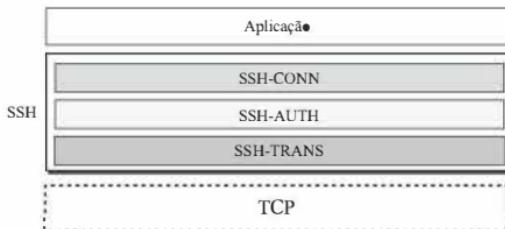



Figura 2.32 Componentes do SSH.

Protocolo SSH *Transport-Layer* (SSH-TRANS)

Como o TCP não é um protocolo da camada de transporte seguro, o SSH primeiro utiliza um protocolo que cria um canal seguro sobre o TCP. Essa nova camada é um protocolo independente conhecido como SSH-TRANS. Quando a rotina que implementa esse protocolo é invocada, o cliente e o servidor primeiro usam o protocolo TCP para estabelecer uma conexão insegura. Em seguida, eles trocam diversos parâmetros de segurança para estabelecer um canal seguro sobre o TCP. Discutiremos a segurança da camada de transporte no Capítulo 1●, mas aqui listamos brevemente os serviços providos por esse protocolo:

1. Privacidade ou confidencialidade das mensagens trocadas
2. Integridade dos dados, o que significa que as mensagens trocadas entre o cliente e o servidor não podem ser alteradas por um intruso sem que a alteração seja percebida
3. Autenticação do servidor, o que significa que o cliente pode ter certeza da autenticidade do servidor
4. Compressão das mensagens, o que melhora a eficiência do sistema e dificulta ataques

Protocolo SSH *Authentication* (SSH-AUTH)

Depois do estabelecimento de um canal seguro entre o cliente e o servidor e do servidor ser autenticado em relação ao cliente, o SSH pode invocar outra rotina que autentica o cliente em relação ao servidor. O processo de autenticação do cliente no SSH é muito semelhante ao que é feito no SSL (Secure Socket Layer), que discutiremos no Capitulo 1●. Essa camada define uma série de ferramentas de autenticação semelhantes àquelas usadas no SSL. A autenticação se inicia com o cliente, que envia uma mensagem de requisição para o servidor. A requisição inclui o nome do usuário, nome do servidor, o método de autenticação e os dados necessários. O servidor responde com uma mensagem de sucesso, o que confirma que o cliente foi autenticado, ou com uma mensagem de erro, o que significa que o processo precisa ser repetido com uma nova mensagem de requisição.

Protocolo SSH *Connection* (SSH-CONN)

Depois que o canal seguro foi estabelecido e tanto servidor como cliente foram mutuamente autenticados, o SSH pode invocar um *software* que implementa o terceiro protocolo, o SSH-CONN. Um dos serviços fornecidos pelo protocolo SSH-CONN é a multiplexação. O SSH-CONN usa o canal seguro estabelecido pelos dois protocolos anteriores e permite que o cliente crie vários canais lógicos sobre ele, e cada um pode ser utilizado para uma finalidade distinta, como acesso remoto, transferência de arquivos, entre outros.

Aplicações

Embora o SSH seja visto muitas vezes como um substituto para o TELNET, é, na verdade, um protocolo de propósito geral que fornece conexão segura entre um cliente e um servidor.

SSH e acesso remoto

Vários aplicativos gratuitos e comerciais usam o SSH para acesso remoto. Entre eles, está o PuTTy, criado por Simon Tatham, que é um programa de SSH cliente que pode ser usado para acesso remoto. Outro aplicativo é o Tectia, que pode ser utilizado em várias plataformas.

SSH para transferência de arquivos

Um dos aplicativos construídos sobre o SSH para transferência de arquivos é o Programa de Transferência de Arquivos Seguro (SFTP – Secure File Transfer Program). O *sftp* utiliza um dos canais fornecidos pelo SSH para transferir arquivos. Outro aplicativo comum é o Cópia Segura (scp – Secure Copy), que utiliza o mesmo formato que o comando de cópia do UNIX, o *cp*, para copiar arquivos.

Encaminhamento de porta

Um dos serviços interessantes prestados pelo protocolo SSH é o chamado **encaminhamento de porta** (*port forwarding*). Podemos utilizar os canais seguros disponíveis no SSH para acessar um aplicativo que não forneça serviços de segurança. Aplicativos como o TELNET e o SMTP podem usar os serviços do mecanismo de encaminhamento de porta do SSH, que cria um túnel pelo qual as mensagens pertencentes a outros protocolos podem transitar. Por isso, esse mecanismo é muitas vezes denominado *tunelamento* SSH. A Figura 2.33 mostra o conceito de encaminhamento de porta para garantir a segurança de uma aplicação FTP.

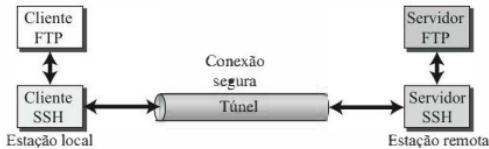



Figura 2.33 Encaminhamento de porta.

O cliente FTP pode usar o cliente SSH na estação local para estabelecer uma conexão segura com o servidor SSH na estação remota. Qualquer solicitação do cliente FTP para o servidor FTP é feita por meio do túnel fornecido pelo cliente e servidor SSH. Qualquer resposta do servidor FTP para o cliente FTP também é feita pelo túnel fornecido pelo cliente e servidor SSH.

Formato dos pacotes SSH

A Figura 2.34 mostra o formato dos pacotes usado pelo protocolo SSH.

O campo de comprimento define o comprimento do pacote, sem incluir o *padding* (*bytes* adicionais inseridos para preenchimento). Um padding de um a oito *bytes* é acrescentado ao pacote para dificultar ataques contra sua segurança. O campo de Verificação de Redundância Cíclica (CRC – Cyclic Redundancy Check) é utilizado para detecção de erros. O campo de tipo designa o tipo de pacote usado em diferentes protocolos SSH. O campo de dados consiste nos dados transferidos pelo pacote em diferentes protocolos.

Figura 2.34 Formato do pacote SSH.

2.3.6 Sistema de Nomes de Domínio

A última aplicação cliente-servidor que discutimos aqui foi concebida para ajudar outros programas da camada de aplicação. Para identificar uma entidade, os protocolos TCP/IP usam o endereço IP, que identifica univocamente a conexão de um *host* com a Internet. No entanto, as pessoas preferem usar nomes em vez de endereços numéricos, portanto, a Internet precisa ter um sistema de diretórios que possa mapear um nome para um endereço. Isto é análogo à rede telefônica. A rede telefônica foi projetada para usar números de telefone, e não nomes. Cada pessoa pode manter uma lista privada mapeando um nome para o número de telefone correspondente, ou pode ligar para o serviço de atendimento para obter tal informação. Aqui, discutimos como o sistema de diretórios na Internet pode mapear nomes para endereços IP.

Como a Internet atualmente é muito grande, um sistema de diretório central não conseguiria suportar todo o mapeamento. Além disso, se o computador central falhasse, a rede de comunicação toda entraria em colapso. Uma solução mais adequada é distribuir a informação entre vários computadores no mundo todo. Nesse método, o *host* que precisa de um mapeamento pode contactar o computador mais próximo que tenha a informação necessária. Esse método é usado pelo **Sistema de Nomes de Domínio** (DNS – Domain Name System). Primeiramente, discutiremos os conceitos e ideias por trás do DNS. A seguir, descreveremos o protocolo DNS propriamente dito.

A Figura 2.35 mostra como o TCP/IP usa um cliente DNS e um servidor DNS para mapear um nome para um endereço. Um usuário quer usar um cliente de transferência de arquivos para acessar o servidor de transferência de arquivos correspondente que está sendo executado em um *host* remoto. O usuário conhece apenas o nome do servidor de transferência de arquivos, tal como *fontedearquivos.com*. No entanto, a pilha de protocolos TCP/IP exige o endereço IP do servidor de transferência de arquivos para estabelecer a conexão. Os seis passos a seguir permitem o mapeamento do nome do *host* para um endereço IP:

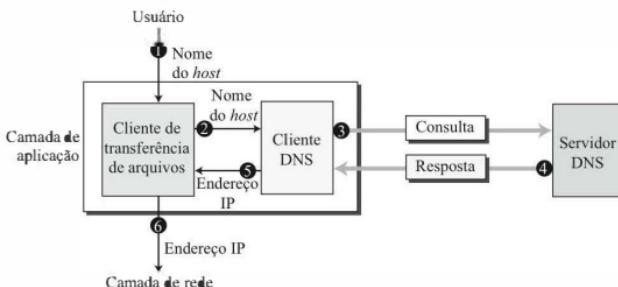



Figura 2.35 Propósito do DNS.

1. O usuário fornece o nome do *host* para o cliente de transferência de arquivos.
2. O cliente de transferência de arquivos passa o nome do *host* para o cliente DNS.
3. Cada computador, depois de inicializado, conhece o endereço de um servidor DNS. O cliente DNS envia uma mensagem para um servidor DNS com uma consulta, na qual consta o nome do servidor de transferência de arquivos, usando o endereço IP (conhecido) daquele servidor DNS.
4. O servidor DNS responde com o endereço IP do servidor de transferência de arquivos desejado.
5. O cliente DNS passa o endereço IP para o servidor de transferência de arquivos.
6. O cliente de transferência de arquivos usa o endereço IP recebido para acessar o servidor de transferência de arquivos.

Perceba que o objetivo de acessar a Internet é criar uma conexão entre o cliente e o servidor de transferência de arquivos, mas antes que isto seja possível, outra conexão precisa ser criada entre o cliente DNS e o servidor DNS, ou seja, precisamos de pelo menos duas conexões nesse caso. A primeira é para mapear o nome para um endereço IP, e a segunda, para a transferência de arquivos. Veremos mais tarde que o mapeamento pode exigir mais de uma conexão.

Espaço de nomes

Para que não haja ambiguidade, os nomes atribuídos às máquinas devem ser cuidadosamente selecionados a partir de um espaço de nomes no qual haja controle completo sobre a ligação entre nomes e endereços IP. Ou seja, os nomes devem ser únicos porque os endereços são únicos. Um **espaço de nomes** (ou *name space*) que mapeia cada endereço para um nome exclusivo pode ser organizado de duas formas: de forma hierárquica ou simples (ou seja, não hierárquica). Em um *espaço de nomes simples*, um nome é atribuído a um endereço, é uma sequência de caracteres sem qualquer estrutura. Os nomes podem ou não ter uma parte em comum; se houver uma parte em comum, entretanto, ela não carrega qualquer significado. A principal desvantagem de um espaço de nomes simples é que ele não pode ser utilizado em um sistema de grandes dimensões, como a Internet, porque deve ser controlado de forma centralizada para evitar ambiguidade e duplicações. Em um *espaço de nomes hierárquico*, cada nome é composto por várias partes. A primeira parte pode definir a natureza da organização, a segunda, o nome dela, a terceira parte pode definir departamentos dentro dela e assim por diante. Nesse caso, a autoridade que atribui e controla os espaços de nomes pode ser descentralizada. Uma autoridade central pode atribuir parte do nome que define a natureza e o nome da organização. A responsabilidade pelo resto do nome pode ser deixada para a organização em si. A organização pode adicionar sufixos (ou prefixos) ao nome para definir o *host* ou recursos. Os gerentes não precisam se preocupar se o prefixo escolhido para um *host* for usado por uma outra organização porque, mesmo se parte de um endereço coincidir, o endereço completo será diferente. Por exemplo, considere que duas organizações nomeiem um de seus computadores de *pedro*. A autoridade central atribui um nome à primeira organização, tal como *primeiro.com*, enquanto a segunda organização recebe o nome *segundo.com*. Quando cada uma delas acrescentar o nome *pedro* ao nome que já lhes foi dado, serão dois nomes distintos: *pedro.primeiro.com* e *pedro.segundo.com*. Os nomes são únicos.

Espaço de nomes de domínio

Para ter um espaço de nomes hierárquico, um espaço de nomes de domínio (*domain name space*) foi projetado. Nesse projeto, os nomes são definidos em uma estrutura de árvore invertida, com a raiz no topo. A árvore pode ter apenas 128 níveis: do nível 0 (raiz) até o nível 127 (ver Figura 2.36).

Rótulo Cada nó da árvore tem um **rótulo** (*label*), que consiste em uma sequência de, no máximo, 63 caracteres. O rótulo-raiz é uma sequência de caracteres nula (uma cadeia vazia). O DNS exige

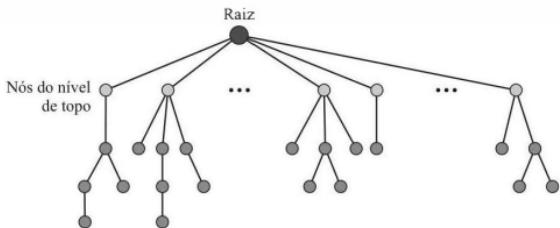



Figura 2.36 Espaço de nomes de domínio.

que os filhos de um nó (os nós que são ramificações do mesmo nó) tenham rótulos diferentes, o que garante a unicidade dos nomes de domínio.

Nome de domínio Cada nó da árvore tem um nome de domínio. Um **nome de domínio** completo é uma sequência de rótulos separados por pontos (.). Os nomes de domínio são sempre lidos a partir do nó local em direção à raiz. O último rótulo é o rótulo-raiz (nulo). Isto significa que um nome de domínio completo sempre termina em um rótulo nulo, e, portanto, que o último caractere é um ponto porque a sequência de caracteres nula é uma cadeia vazia. A Figura 2.37 mostra alguns nomes de domínio.

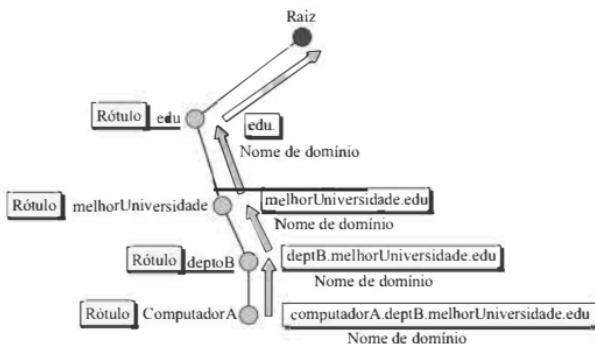



Figura 2.37 Nomes de domínio e rótulos.

Se um rótulo termina com uma cadeia de caracteres nula, ele é denominado **Nome de Domínio Totalmente Qualificado** (FQDN – Fully Qualified Domain Name). O nome deve terminar com um rótulo nulo, mas como nulo significa vazio, o rótulo termina com um ponto. Se um rótulo não terminar com uma cadeia vazia, ele é denominado **Nome de Domínio Parcialmente Qualificado** (PQDN – Partially Qualified Domain Name). Um PQDN se inicia em um nó, mas não atinge a raiz. Ele é utilizado quando o nome a ser resolvido pertence ao mesmo local que o cliente. Nesse caso, a entidade que realiza a resolução pode fornecer a parte que falta, chamada de *sufixo*, para criar um FQDN.

Domínio

Um domínio é uma subárvore do espaço de nomes de domínio. O nome do domínio é o nome do nó no topo da subárvore. A Figura 2.38 mostra alguns domínios. Perceba que um domínio pode ser dividido em domínios.

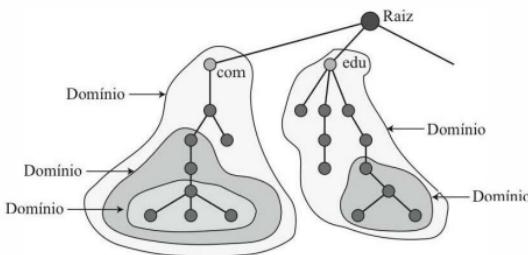



Figura 2.38 Domínios

Distribuição do espaço de nomes

A informação contida no espaço de nomes de domínio deve ser armazenada. No entanto, seria muito ineficiente e também pouco confiável ter apenas um computador armazenando uma quantidade tão grande de informações, pois responder a consultas vindas do mundo todo sobrecarregaria o sistema. Não seria confiável porque qualquer falha tornaria os dados inacessíveis.

Hierarquia de servidores de nomes A solução para esses problema é distribuir a informação entre diversos computadores chamados **servidores DNS**. Uma maneira de fazer isso é dividir o espaço completo em vários domínios com base no primeiro nível. Em outras palavras, deixamos a raiz sozinha e criamos tantos domínios (subárvores) quanto existam nós no primeiro nível. Como um domínio criado desta forma poderia ser muito grande, o DNS permite que os domínios sejam subdivididos em domínios ainda menores (subdomínios). Cada servidor (denominado autoritativo) pode ficar responsável por um domínio grande ou pequeno. Em outras palavras, temos uma hierarquia de servidores da mesma maneira que temos uma hierarquia de nomes (ver Figura 2.39).

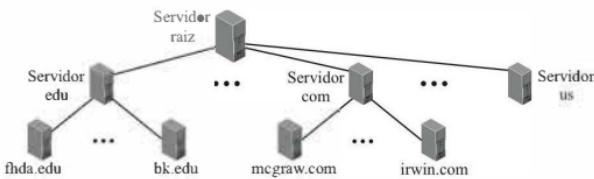



Figura 2.39 Hierarquia de servidores de nomes.

Zona

Como a hierarquia completa de nomes de domínio não pode ser armazenada em um único servidor, ela é dividida entre vários servidores. Aquilo sobre o qual um servidor tem responsa-

bilidade ou autoridade é chamado zona. Podemos definir uma zona como uma parte contígua da árvore toda. Se um servidor aceita a responsabilidade para um domínio e não o divide em domínios menores, o "domínio" e a "zona" referem-se à mesma coisa. Nesse caso, o servidor cria um banco de dados chamado arquivo de zona e mantém todas as informações sobre cada nó dentro desse domínio. No entanto, se um servidor dividir seu domínio em subdomínios e delegar parte de sua autoridade a outros servidores, o "domínio" e a "zona" se referem a coisas distintas. As informações sobre os nós dos subdomínios são armazenadas nos servidores em níveis mais baixos, sendo que o servidor original mantém algum tipo de referência aos de nível inferior. Claro que o original não se livra totalmente da responsabilidade. Ele ainda tem uma zona, mas as informações detalhadas são mantidas pelos servidores dos níveis inferiores (ver Figura 2.40).

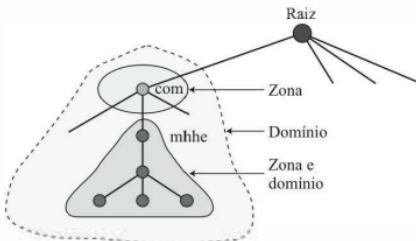



Figura 2.40 Zona

Servidor-raiz

Um **servidor-raiz** é um servidor cuja zona consiste na árvore inteira; ele normalmente não armazena qualquer informação sobre domínios, mas delega sua autoridade a outros servidores, mantendo referências a eles. Existem vários servidores-raiz, cada um cobrindo o espaço de nomes de domínio completo. Os servidores-raiz são distribuídos pelo mundo todo.

Servidores primários e secundários O DNS define dois tipos de servidores: primários e secundários. Um *servidor primário* armazena um arquivo relativo à zona sobre a qual ele tem autoridade. É responsável pela criação, manutenção e atualização do arquivo de zona, que mantém armazenado em seu disco local.

Um *servidor secundário* recebe todas as informações sobre uma zona de um outro servidor (primário ou secundário) e armazena tal arquivo em seu disco local. O servidor secundário não cria nem atualiza os arquivos de zona. Se uma atualização for necessária, ela deve ser feita pelo servidor primário, que envia a versão atualizada para o secundário.

Tanto os servidores primários como os secundários têm autoridade sobre as zonas que servem. A ideia não é colocar o servidor secundário em um nível de autoridade inferior, mas sim, criar redundância de dados de modo que, se um servidor falhar, o outro pode continuar a servir os clientes. Perceba também que um servidor pode ser primário para uma zona específica e secundário para outra. Portanto, quando dizemos que um servidor é primário ou secundário, devemos ter cuidado com a zona à qual nos referimos.

Um servidor primário obtém todas as informações do arquivo em disco; o servidor secundário obtém todas as informações do servidor primário.

DNS na Internet

O DNS é um protocolo que pode ser usado em diferentes plataformas. Na Internet, o espaço de nomes de domínio (árvore) foi originalmente dividido em três seções distintas: domínios genéricos, de país e reversos. No entanto, devido ao rápido crescimento da Internet, tornou-se extremamente difícil manter o controle dos domínios reversos, que poderiam ser usados para encontrar o nome de um *host* a partir do seu endereço IP. Os domínios reversos são agora obsoletos (ver RFC 3425). Portanto, vamos nos concentrar nos dois primeiros.

Domínios genéricos

Os **domínios genéricos** definem *hosts* registrados de acordo com o seu comportamento genérico. Cada nó na árvore define um domínio, que é um índice para a base de dados do espaço de nomes de domínio (ver Figura 2.41).

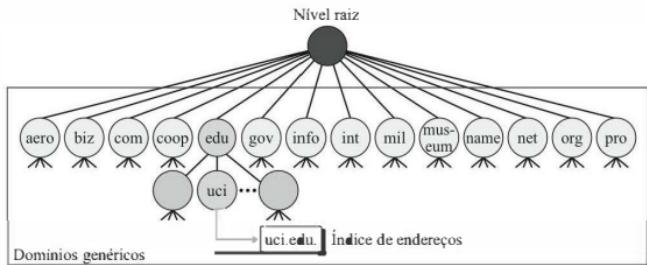



Figura 2.41 Domínios genéricos.

Observando a árvore, vemos que o primeiro nível na seção de domínios genéricos admite 14 rótulos possíveis, que descrevem os diversos tipos de organização, conforme listado na Tabela 2.12.

Tabela 2.12 Rótulos de domínio genéricos.

Rótulo	Descrição	Rótulo	Descrição
aero	Linhas aéreas e setor aeroespacial	**int**	Organizações internacionais
biz	Empresas ou firmas	**mil**	Grupos militares
com	Organizações comerciais	**museum**	Museus
coop	Cooperativas	**name**	Nomes pessoais (indivíduos)
edu	Instituições educacionais	**net**	Provedores de rede
gov	Instituições governamentais	**org**	Organizações sem fins lucrativos
info	Provedores de serviços de informação	**pro**	Organizações profissionais

Domínios de país

A seção de **domínios de país** usa abreviaturas de dois caracteres para indicar o país (por exemplo, *us* para os Estados Unidos e *br* para o Brasil). O segundo rótulo pode ser organizacional, ou

uma denominação mais específica, de abrangência nacional. Os Estados Unidos, por exemplo, utilizam abreviações do estado como uma subdivisão do *us* (por exemplo, *ca.us.*). A Figura 2.42 mostra a seção de domínios de país. O endereço *uci.ca.us.* pode ser traduzido como Universidade da Califórnia, uma universidade da cidade de Irvine, no estado da Califórnia, nos Estados Unidos.

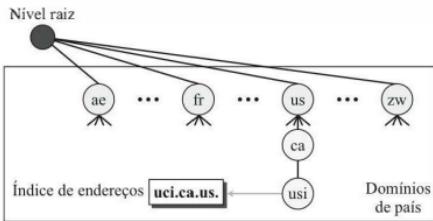



Figura 2.42 Domínios de país.

Resolução

O processo de mapear um nome para um endereço é chamado *resolução de nome-endereço*. O DNS foi projetado como uma aplicação cliente-servidor. Um *host* que precisa mapear um endereço para um nome, ou um nome para um endereço, invoca um cliente DNS chamado **resolvedor**, que acessa o servidor DNS mais próximo com um pedido de mapeamento. Se o servidor tiver a informação, ele responde ao resolvedor; caso contrário, pode redirecionar o resolvedor para outros servidores ou pedir a estes que forneçam a informação requisitada. Após o resolvedor receber o resultado do mapeamento, ele interpreta a resposta para ver se a resolução foi bem-sucedida ou se houve algum erro, e finalmente devolve o resultado para o processo que o solicitou. A resolução pode ser recursiva ou iterativa.

Resolução recursiva

A Figura 2.43 mostra um exemplo simples de uma resolução recursiva. Consideremos que um aplicativo sendo executado em uma estação chamada *algum.anet.com* precisa encontrar o endereço IP de outra estação chamada *engenharia.mcgraw-hill*.com para enviar uma mensagem. A estação de origem está conectada ao ISP Anet; a estação de destino está conectada à rede McGraw-Hill.

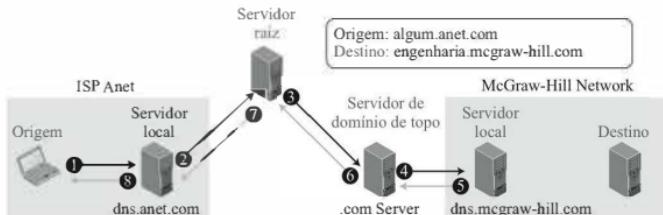



Figura 2.43 Resolução recursiva.

O aplicativo na estação de origem invoca o resolvedor DNS (cliente) para localizar o endereço IP da estação de destino. O resolvedor, que não conhece o endereço, envia a consulta para o servidor de DNS local (por exemplo, *dns.anet.com*) sendo executado no lado do ISP Anet (evento 1). Considere que esse servidor também não sabe o endereço IP da estação de destino. Ele envia a consulta para um servidor de DNS raiz, cujo endereço IP é supostamente conhecido pelo servidor de DNS local (evento 2). Os servidores-raiz normalmente não mantêm os mapeamentos entre nomes e endereços IP, mas um servidor-raiz deve conhecer pelo menos um servidor em cada domínio de topo (*top-level domain*); neste caso, ele deve conhecer o servidor responsável pelo domínio *com*. A consulta é enviada para este servidor de domínio de topo (evento 3). Considere que o servidor não sabe o mapeamento nome-endereço do destino específico, mas sabe o endereço IP do servidor de DNS local na companhia McGraw-Hill (por exemplo, *dns.mcgraw-hill.com*). A consulta é enviada para o servidor (evento 4), que conhece o endereço IP da estação de destino, agora enviado de volta ao servidor DNS de domínio de topo (evento 5), em seguida de volta ao servidor-raiz (evento 6), e então de volta ao servidor DNS do ISP, que pode armazenar esse mapeamento em *cache* para as futuras consultas (evento 7) e finalmente de volta à estação de origem (evento 8).

Resolução iterativa

Na resolução iterativa, cada servidor que não sabe o mapeamento envia o endereço IP do próximo servidor de volta àquele que solicitou o mapeamento. A Figura 2.44 mostra o fluxo de informações em uma resolução iterativa no mesmo cenário ilustrado na Figura 2.43. Normalmente, a resolução iterativa ocorre entre dois servidores locais; o resolvedor original obtém a resposta final do servidor local. Perceba que as mensagens mostradas nos eventos 2, 4 e 6 contêm a mesma consulta. No entanto, a mensagem mostrada no evento 3 contém o endereço IP do servidor de domínio de topo, a do evento 5 contém o endereço IP do servidor DNS local da McGraw-Hill e a do evento 7 contém o endereço IP do destino. Quando o servidor DNS local da Anet recebe o endereço IP do destino, ele envia o resultado para o resolvedor (evento 8).

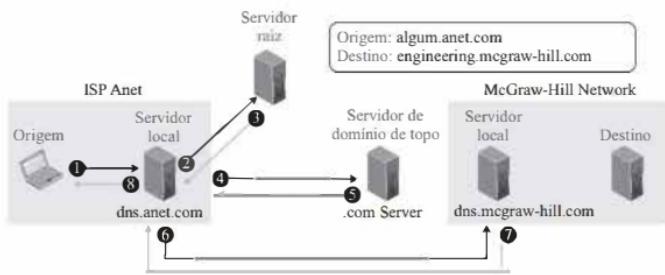



Figura 2.44 Resolução iterativa.

Armazenamento temporário

Cada vez que um servidor recebe uma consulta referente a um nome que não está no seu domínio, ele precisa procurar no banco de dados o endereço IP de um servidor. A redução do intervalo necessário para essa pesquisa levaria a uma maior eficiência. O DNS lida com isto usando um mecanismo de armazenamento temporário (*caching*). Quando um servidor requisita um mapeamento a outro servidor e recebe uma resposta, ele armazena a informação em sua memória de armazenamento temporário (*cache*) antes de enviá-la ao cliente. Se algum cliente pedir o mesmo mapeamento, o

servidor pode verificar a memória *cache* e resolver o problema. No entanto, para informar ao cliente que a resposta está vindo da memória *cache* e não de uma fonte autoritativa, o servidor marca a resposta como *não autoritativa*.

O *caching* acelera a resolução, mas também pode ser problemático. Se um servidor armazena um mapeamento por um longo tempo, pode enviar um mapeamento desatualizado para o cliente. Para evitar isso, duas técnicas são utilizadas. Primeiramente, o servidor autoritativo sempre adiciona uma informação ao mapeamento chamada *tempo de vida* (TTL – Time to live). O TTL define o tempo em segundos que o servidor recebendo a informação pode armazená-la em *cache*. Após esse período, o mapeamento torna-se inválido e qualquer consulta deve ser enviada novamente ao servidor autoritativo. Em segundo lugar, o DNS exige que cada servidor mantenha um contador TTL para cada mapeamento que ele armazena em *cache*. A memória *cache* deve ser verificada periodicamente e os mapeamentos com um TTL expirado devem ser removidos.

Registros de recursos

A informação de zona associada com um servidor é implementada como um conjunto de registros de recursos (*resource records*), ou seja, um servidor de nomes armazena uma base de dados de registros de recursos. Um *registro de recurso* é uma 5-tupla, conforme mostrado a seguir:

(Nome de domínio, Tipo, Classe, TTL, Valor)

O campo de nome de domínio é o que identifica o registro de recurso. O valor define as informações mantidas sobre o nome de domínio. O TTL define o número de segundos durante o qual a informação é válida. A classe define o tipo de rede; estamos interessados apenas na classe IN (Internet). O tipo define como o valor deve ser interpretado. A Tabela 2.13 lista os tipos mais comuns e como o valor é interpretado para cada tipo.

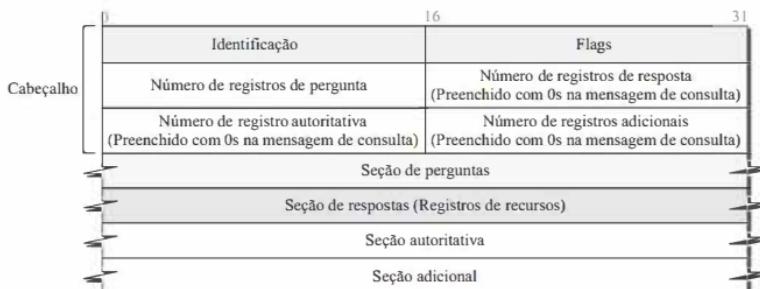



Tabela 2.13 Tipos

Tipo	Interpretação do valor
A	Endereço IPv4 de 32 *bits* (ver Capítulo 4)
NS	Identifica os servidores autoritativos para uma zona
CNAME	Define um pseudônimo (alias) para o nome oficial de um *host*
SOA	Marca o início de uma zona
MX	Redireciona e-mail para um servidor
AAAA	Um endereço IPv6 (ver Capitulo 4)

Mensagens DNS

Para recuperar informações sobre *hosts*, o DNS usa dois tipos de mensagens: *consulta* e *resposta*. Ambos têm o mesmo formato conforme mostra a Figura 2.45.

Vamos discutir brevemente os campos de uma mensagem DNS. O campo de identificação é usado pelo cliente para ligar a resposta à consulta. O campo de identificadores (*flags*) define se a mensagem é uma consulta ou resposta. Ele também inclui os estados de erro. Os próximos quatro campos no cabeçalho definem o número de cada tipo de registro na mensagem. A seção de pergunta, que é incluída na mensagem de consulta e repetida na de resposta, consiste em um ou mais registros de pergunta. Ela está presente tanto na mensagem de consulta como na de resposta. A seção

Figura 2.45 Mensagem DNS.

de resposta consiste em um ou mais registros de recursos. Ela está presente apenas em mensagens de resposta. A seção autoritativa provê informações (nome de domínio) sobre um ou mais servidores autoritativos para a consulta. A seção de informações adicionais fornece dados que podem ajudar o resolvedor.

Exemplo 2.14

No UNIX e no Windows, o utilitário nslookup* pode ser usado para encontrar o mapeamento endereço/nome. A seguir, mostramos como podemos obter um endereço quando o nome de domínio é fornecido.

```
$ nslookup www.forouzan.biz
Nome: www.forouzan.biz
Endereço: 198.170.240.179
```

Encapsulamento

O DNS pode usar os serviços tanto do UDP como do TCP. Em ambos os casos, a porta bem conhecida usada pelo servidor é a 53. O UDP é usado quando o tamanho da mensagem de resposta é inferior a 512 *bytes*, porque a maioria dos pacotes UDP tem um limite de tamanho de 512 *bytes*. Se o tamanho da mensagem de resposta for maior do que 512 *bytes*, uma conexão TCP é usada. Neste caso, pode ocorrer um dos dois cenários:

- Se o resolvedor sabe previamente que o tamanho da mensagem de resposta é maior do que 512 *bytes*, ele usa a conexão TCP. Por exemplo, se um servidor de nomes secundário (atuando como um cliente) precisa das informações de zona vindas de um servidor primário, ele usa uma conexão TCP porque o tamanho da informação a ser transferida geralmente excede 512 *bytes*.

* N. de T.: O utilitário nslookup entrou em desuso nos sistemas UNIX, sendo substituído pelo dig.

- Se o resolvedor não sabe o tamanho da mensagem de resposta, pode usar a porta UDP. No entanto, se o tamanho da mensagem de resposta for maior do que 512 *bytes*, o servidor trunca a mensagem e ativa o *bit* de TC. O resolvedor, então, abre uma conexão TCP e repete o pedido para obter uma resposta completa do servidor.

Registradores de domínios

Como um novo domínio é adicionado ao DNS? Isto é feito por meio de um *registrador de domínios*, uma entidade comercial credenciada pela ICANN. Um registrador primeiro verifica se o nome de domínio solicitado é único e então o adiciona à base de dados do DNS. Uma taxa é cobrada pelo serviço. Existem, atualmente, muitos registradores de domínios. Seus nomes e endereços podem ser encontrados em

http://www.intenic.net

Para se registrar, a organização precisa fornecer o nome e o endereço IP do servidor. Por exemplo, uma nova organização comercial chamada *maravilha* tendo um servidor chamado *ws*, cujo endereço IP é 200.200.200.5, precisa fornecer a seguinte informação para um dos registradores:

Nome de domínio: ws.maravilha.com   **Endereço IP:** 200.200.200.5

DDNS

Quando o DNS foi concebido, ninguém previu que haveria tantas mudanças de endereços. No DNS, quando há uma mudança, como a adição ou remoção de um *host*, ou a mudança de um endereço IP, a alteração deve ser feita no arquivo-mestre do DNS. Esses tipos de mudanças envolvem um grande número de atualizações manuais. O tamanho da Internet atual não permite esse tipo de operação manual.

O arquivo-mestre do DNS deve ser atualizado dinamicamente. O **Sistema de Nomes de Domínio Dinâmico** (DDNS – Dynamic Domain Name System), foi, portanto, concebido para responder a essa necessidade. No DDNS, quando uma ligação entre um nome e um endereço é definida, a informação é enviada, geralmente via DHCP (discutido no Capítulo 4) para um servidor DNS primário. O servidor primário atualiza a zona. Os servidores secundários são notificados ativa ou passivamente. Na notificação ativa, o servidor primário envia uma mensagem para os servidores secundários sobre a mudança na zona, enquanto na notificação passiva os servidores secundários verificam periodicamente quaisquer alterações. Em ambos os casos, depois de ter sido notificado sobre a mudança, o servidor secundário solicita informações sobre toda a zona (a chamada transferência de zona).

Para prover a segurança e impedir alterações não autorizadas nos registros de DNS, o DDNS pode usar algum mecanismo de autenticação.

Segurança do DNS

O DNS é um dos sistemas mais importantes na infraestrutura da Internet, pois fornece serviços cruciais para seus usuários. Aplicativos como o acesso à Web ou e-mail dependem muito do correto funcionamento do DNS, que pode ser atacado de diversas formas, incluindo:

1. O atacante pode ler a resposta de um servidor DNS para descobrir a natureza ou nomes de *sites* que o usuário acessa com maior frequência. Esse tipo de informação pode ser usado para montar um perfil do usuário. Para evitar esse ataque, as mensagens DNS precisam ser confidenciais (ver Capítulo 10).

2. O atacante pode interceptar a resposta de um servidor DNS e alterá-la ou criar uma nova resposta totalmente falsa com o intuito de direcionar o usuário para um *site* ou domínio de sua escolha. Pode-se prevenir esse tipo de ataque por meio da autenticação da origem da mensagem e verificação de sua integridade (ver Capítulo 10).
3. O atacante pode inundar o servidor DNS com requisições com o objetivo de sobrecarregá--lo ou até derrubá-lo. Esse tipo de ataque pode ser evitado usando algum mecanismo contra ataques de negação de serviço.

Para proteger o DNS, a IETF desenvolveu uma tecnologia chamada *DNS Security* (DNSSEC – Segurança do DNS), que fornece autenticação da origem e integridade da mensagem usando um serviço de segurança denominado *assinatura digital* (ver Capítulo 10). O DNSSEC, no entanto, não fornece confidencialidade para as mensagens DNS. Não há proteção especifica contra ataques de negação de serviço na especificação do DNSSEC, mas, o sistema de *cache*, até certo ponto, protege os servidores de nível mais alto contra estes ataques.

2.4 PARADIGMA *PEER-TO-PEER*

Discutimos o paradigma cliente-servidor no início do capítulo. Discutimos também algumas aplicações cliente-servidor padronizadas nas seções anteriores. Nesta seção, discutimos o paradigma *peer-to-peer* (também conhecido como par a par). O primeiro exemplo de compartilhamento de arquivos via *peer-to-peer* remonta a dezembro de 1987, quando Wayne Sino criou o *WWIVnet*, o componente de rede do Guerra Mundial Quatro (WWIV – World War Four), um Sistema de Quadro de Avisos (também conhecido como *Bulletin Board System*). Em julho de 1999, Ian Clarke projetou a Freenet, um sistema de armazenamento de dados descentralizado, distribuído e resistente a censuras, com o objetivo de fornecer liberdade de expressão através de uma rede *peer-to-peer* com forte proteção de anonimato.

O paradigma *peer-to-peer* ganhou popularidade com o Napster (1999-2001), um serviço de compartilhamento de arquivos de música *online* criado por Shawn Fanning. Embora a cópia e distribuição livre de arquivos de música pelos usuários tenham levado a uma ação judicial contra o Napster por violação de direitos autorais, o que causou o fechamento do serviço, ele abriu o caminho para modelos de distribuição de arquivos baseados em *peer-to-peer* que surgiram mais tarde. O Gnutella foi lançado em março de 2000, seguido pelo Fast-Track (usado pelo Kazaa), BitTorrent, WinMX e GNUnet em março, abril, maio e novembro de 2001, respectivamente.

2.4.1 Redes P2P

Usuários da Internet que queiram compartilhar os seus recursos tornam-se *peers* e formam uma rede. Quando um *peer* na rede tem um arquivo (por exemplo, um arquivo de áudio ou de vídeo) para compartilhar, ele o deixa disponível para o restante dos *peers*. Um *peer* interessado pode se conectar ao computador onde o arquivo está armazenado e fazer o *download*. Depois de um *peer* obter um arquivo, ele pode deixá-lo disponível para outros *peers* para *download*. À medida que os *peers* se juntam à rede e obtêm o arquivo, mais cópias passam a ficar disponíveis para o grupo. Como as listas de *peers* podem crescer ou diminuir, a questão é como o paradigma rastreia os *peers* leais e a localização dos arquivos. Para responder a estas perguntas, precisamos primeiro dividir as redes P2P em duas categorias: centralizadas e descentralizadas.

Redes centralizadas

Em uma rede P2P centralizada, o sistema de diretórios – a lista dos *peers* e do que eles têm a oferecer – usa o paradigma cliente-servidor, mas o armazenamento e o *download* dos arquivos são feitos

usando o paradigma *peer-to-peer*. Assim, uma rede P2P centralizada é muitas vezes denominada rede P2P híbrida. O Napster foi um exemplo de uma rede P2P centralizada. Nesse tipo de rede, um *peer* primeiro se registra junto a um servidor central. Em seguida, o *peer* fornece o seu endereço IP e uma lista de arquivos que ele tem para compartilhar. Para evitar o colapso do sistema, o Napster usava vários servidores para isso, apesar de mostrarmos apenas um na Figura 2.46.

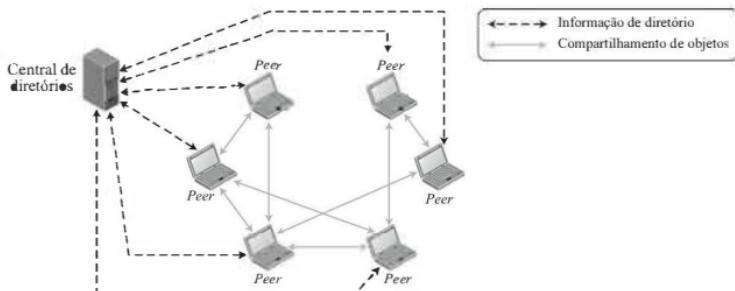



Figura 2.46 Rede centralizada.

Um *peer*, ao procurar um determinado arquivo, envia uma consulta para um servidor central. O servidor procura em seu diretório e responde com os endereços IP dos nós que têm uma cópia do arquivo. O *peer* entra em contato com um dos nós e faz o *download* do arquivo. O diretório é constantemente atualizado à medida que os nós entram ou saem da rede.

Redes centralizadas simplificam a manutenção dos diretórios, mas apresentam vários inconvenientes. O acesso ao diretório pode gerar uma quantidade enorme de tráfego e deixar o sistema lento. Os servidores centrais são vulneráveis a ataques, e se todos eles falharem, o sistema inteiro deixa de funcionar. O componente central do sistema foi o principal responsável pelo Napster perder a ação judicial por violação de direitos autorais e pelo seu fechamento em julho de 2001. A Roxio trouxe de volta o Novo Napster em 2003; o Napster versão 2 é agora um *site* legal para venda de músicas.

Rede descentralizada

Uma rede P2P descentralizada não depende de um sistema de diretórios centralizado. Nesse modelo, os *peers* se organizam em uma rede sobreposta (*overlay network*), uma rede lógica criada sobre uma rede física. Dependendo de como os nós da rede sobreposta são ligados, uma rede P2P descentralizada pode ser classificada como não estruturada ou estruturada.

Redes não estruturadas

Em uma rede P2P não estruturada, os nós se ligam de forma aleatória. Uma busca em uma rede P2P não estruturada não é muito eficiente, pois uma consulta para localizar um arquivo deve inundar toda a rede (ou seja, ser enviada a todos os nós), o que produz uma quantidade significativa de tráfego e pode não retornar resultados. Dois exemplos desse tipo de rede são Gnutella e Freenet. A seguir, discutimos a rede Gnutella como um exemplo.

Gnutella A rede Gnutella é um exemplo de uma rede *peer-to-peer* descentralizada, porém não estruturada, no sentido de que o diretório é aleatoriamente distribuído entre os nós. Quando o nó A quer acessar um objeto (tal como um arquivo), ele entra em contato com um de seus vizinhos.

Um vizinho, nesse caso, é qualquer nó cujo endereço seja conhecido pelo nó A. O nó A envia uma mensagem de *consulta* ao seu vizinho, o nó W. A consulta inclui o identificador do objeto (por exemplo, o nome do arquivo). Se o nó W sabe o endereço do nó X, que tem o objeto, ele envia uma mensagem de *resposta*, a qual inclui o endereço do nó X. O nó A agora pode usar os comandos definidos em um protocolo de transferência, como HTTP, para obter do nó X uma cópia do objeto. Se o nó W não sabe o endereço do nó X, ele *inunda* o pedido de A para todos os seus vizinhos. Por fim, um dos nós da rede responde à mensagem de *consulta*, e o nó A ganha acesso ao nó X. Discutiremos o mecanismo de inundação no Capítulo 4, quando discutimos protocolos de roteamento, mas é importante mencionar aqui que, embora as inundações no Gnutella sejam de certa forma controladas para evitar elevadas cargas de tráfego, uma das razões pelas quais o Gnutella apresenta baixa escalabilidade é o mecanismo de inundação.

Uma das perguntas que ainda precisam ser respondidas é como, de acordo com o processo descrito anteriormente, o nó A sabe o endereço de pelo menos um vizinho. Isto é feito no momento da inicialização (*bootstrap*), quando o nó instala o *software* Gnutella pela primeira vez. O *software* inclui uma lista de nós (*peers*) que o nó A pode registrar como vizinhos. Mais tarde, o nó A pode usar duas mensagens, chamadas *ping* e *pong*, para tentar descobrir se um vizinho ainda está ativo.

Conforme mencionado anteriormente, um dos problemas com a rede Gnutella é a falta de escalabilidade causada pelo mecanismo de inundação. Quando o número de nós aumenta, as inundações não levam a um bom resultado. Para tornar a consulta mais eficiente, uma nova versão do Gnutella implementou um sistema hierárquico de *folhas* e *ultranós*. Um nó entrante na rede torna-se uma folha, não se responsabilizando pelo roteamento; nós que são capazes de auxiliar no roteamento são promovidos a ultranós. Isto permite que as consultas propaguem-se até mais longe, aumentando a eficiência e a escalabilidade do sistema. O Gnutella adotou uma série de outras técnicas, como a inclusão do Protocolo de Roteamento de Consulta (QRP – Query Routing Protocol) e da Consulta Dinâmica (DQ – Dynamic Querying) para reduzir a carga de tráfego e tornar as pesquisas mais eficientes.

Redes estruturadas

Uma rede estruturada utiliza um conjunto predefinido de regras para ligar nós para que as consultas sejam efetiva e eficientemente resolvidas. A técnica mais comum utilizada com essa finalidade é a Tabela de *Hash* Distribuída (DHT – Distributed Hash Table). A DHT é usada em muitas aplicações, incluindo Estrutura de Dados Distribuída (DDS – Distributed Data Structure), Sistemas de Conteúdos Distribuídos (CDS – Content Distributed Systems), Sistema de Nomes de Domínio (DNS – Domain Name System) e compartilhamento de arquivos P2P. Um protocolo de compartilhamento de arquivos P2P popular que usa DHT é o BitTorrent. Discutiremos o conceito de DHT independentemente na próxima seção como uma técnica que pode ser utilizada tanto em redes P2P estruturadas como em outros sistemas.

2.4.2 Tabela de *Hash* Distribuída

Uma **Tabela de *Hash* Distribuída** (DHT – Distributed Hash Table) distribui dados (ou referências a dados) entre um conjunto de nós de acordo com algumas regras predefinidas. Cada *peer* em uma rede baseada em DHT torna-se responsável por uma série de dados. Para evitar a sobrecarga causada pelas inundações que discutimos no caso das redes P2P não estruturadas, redes baseadas em DHT permitem que cada *peer* tenha conhecimento parcial sobre a rede toda. Este conhecimento pode ser utilizado para rotear as consultas por itens até os nós responsáveis por eles, usando procedimentos eficazes e escaláveis que iremos discutir em breve.

Espaço de endereços

Em uma rede baseada em DHT, cada *peer* e cada conjunto de dados é mapeado para um ponto em um grande espaço de endereços de tamanho 2^m. O espaço de endereços é concebido usando

aritmética modular, o que significa que podemos pensar nos pontos do espaço de endereços como se eles fossem distribuídos uniformemente sobre um círculo com 2^m pontos (0 a $2^m - 1$), em sentido horário, conforme apresentado na Figura 2.47. A maioria das implementações de DHT usa $m = 160$.

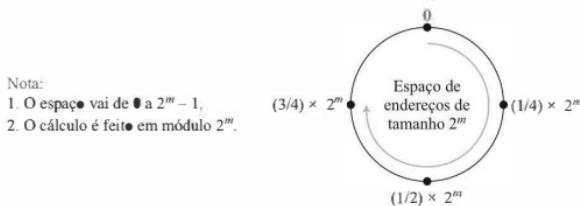



Figura 2.47 Espaço de endereços.

Calculando o *hash* do identificador do *peer*

O primeiro passo na criação do sistema DHT é colocar todos os *peers* sobre o anel que representa o espaço de endereços. Para isso, é usada uma função que calcula o *hash* do identificador do *peer*, normalmente seu endereço IP, mapeando-o para um inteiro de *m bits* chamado de ***ID de nó***.

ID do nó = *hash* (Endereço IP do *peer*)

Uma função de *hash* é uma função matemática que cria uma saída a partir de uma entrada. No entanto, a DHT utiliza alguma das funções de *hash* criptográficas, como os algoritmos da família Algoritmo de *Hash* Seguro (SHA – Secure Hash Algorithm) que são resistentes à colisão, o que significa que a probabilidade de duas entradas serem mapeadas para a mesma saída é muito baixa. Discutiremos esses algoritmos de *hash* no Capítulo 10.

Calculando o *hash* do identificador do objeto

A função de *hash* também é aplicada sobre o nome do objeto (por exemplo, um arquivo) a ser compartilhado, criando um inteiro de *m bits* no mesmo espaço de endereços. O resultado, no jargão da DHT, é chamado *chave*.

chave = *hash* (Nome do objeto)

Na DHT, um objeto costuma ser identificado pelo par (chave, valor) no qual a chave é o *hash* do nome do objeto e o valor é o objeto ou uma referência ao objeto.

Armazenando o objeto

Existem duas estratégias para armazenar um objeto: o método direto e o indireto. No método direto, o objeto é armazenado no nó cujo ID seja de alguma forma o *mais próximo* à chave no anel. O termo *mais próximo* é definido de forma diferente em cada protocolo. Essa estratégia muito provavelmente envolve a transferência do objeto, que é movido do computador onde originalmente se encontrava. No entanto, a maioria dos sistemas de DHT utiliza o método indireto devido à sua maior eficiência. O *peer* que possui o objeto conserva esse objeto, mas uma referência para ele é criada e armazenada no nó cujo ID é o mais próximo ao ponto correspondente a sua chave. Ou seja, o objeto físico e a referência ao objeto são armazenados em dois locais distintos. Na estratégia direta, criamos um relacionamento entre o ID do nó que armazena o objeto e a chave dele; na indireta, criamos um relacionamento entre a referência (ponteiro) para o objeto e o nó que a armazena. Em ambos os casos, algum relacionamento é necessário para encontrar o objeto quando o nome do objeto é fornecido. Usaremos o método indireto no restante desta seção.

Exemplo 2.15

Embora o valor normal de *m* seja 160, para efeitos de demonstração, usamos *m* = 5 para facilitar o tratamento dos nossos exemplos. Na Figura 2.48, consideramos que vários *peers* já se juntaram ao grupo. O nó N5, cujo endereço IP é 110.34.56.20, tem um arquivo chamado Liberdade que quer compartilhar com seus pares. O nó calcula o *hash* do nome do arquivo, "Liberdade", para obter a chave = 14. Como o nó *mais próximo* à chave 14 é o nó N17, o nó N5 cria uma referência para o nome do arquivo (chave), seu endereço IP, e o número da porta (e possivelmente algumas outras informações sobre o arquivo) e envia tal referência para ser armazenada no nó N17. Em outras palavras, o arquivo é armazenado em N5, a chave do arquivo é C14 (um ponto no anel DHT), mas a referência para o arquivo é armazenada no nó N17. Veremos mais adiante como os outros nós podem primeiramente encontrar N17, extrair a referência e então usá-la para acessar o arquivo Liberdade. Nosso exemplo mostra apenas uma chave no anel, mas em uma situação real, há milhões de chaves e nós no anel.

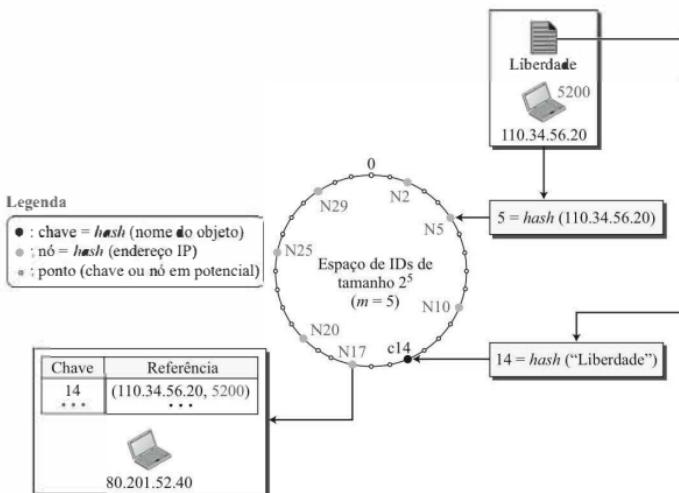



Figura 2.48 Esquema para o Exemplo 2.15.

Roteamento

A função principal da DHT é rotear uma consulta até o nó responsável por armazenar a referência a um objeto. Cada implementação da DHT usa uma estratégia diferente para o roteamento, mas todas seguem a ideia de que cada nó precisa ter um conhecimento parcial sobre o anel para rotear uma consulta para um nó que esteja mais próximo do responsável.

Chegada e saída de nós

Em uma rede P2P, cada *peer* pode ser um *desktop* ou um *laptop*, ligado ou desligado. Quando um *peer* executa o *software* da DHT, ele entra na rede; quando o computador é desligado ou o *peer* fecha o *software*, ele sai da rede. Uma implementação de DHT precisa estabelecer uma estratégia clara e

eficiente para lidar com a chegada e partida dos nós e com o efeito disso sobre o restante dos *peers*. A maioria das implementações de DHT trata a falha de um nó como se ele tivesse saido da rede.

2.4.3 Chord

Existem vários protocolos que implementam sistemas de DHT. Nesta seção, apresentamos três deles: Chord, Pastry e Kademlia. Selecionamos o protocolo Chord por sua simplicidade e sua abordagem elegante para o roteamento de consultas. Em seguida, discutimos o protocolo Pastry porque ele usa uma abordagem diferente do Chord e sua estratégia de roteamento é muito próxima daquela do protocolo Kademlia, que é usado na rede de compartilhamento de arquivos mais popular atualmente, o BitTorrent.

O **Chord** foi publicado por Stoica *et al.*, em 2001. A seguir, analisaremos brevemente as principais características desse algoritmo.

Espaço de identificadores

Dados e nós no Chord são representados por identificadores de *m bits* que criam um espaço de identificadores contendo 2^m pontos distribuídos em um círculo no sentido horário. Referimo-nos à identificação de um dado como *c* (de *chave*) e ao identificador de um *peer* como N (de *nó*). A aritmética no espaço é feita em módulo 2^m, o que significa que os identificadores são cíclicos, indo até $2^m - 1$ e de volta a 0. Embora algumas implementações usem uma função de *hash* resistente a colisões, como o SHA1 com $m = 160$, aqui usamos $m = 5$ para simplificar a discussão. O *peer* mais próximo para o qual $N \geq c$ é chamado de sucessor de c e armazena o valor (c, v), onde c é a chave (*hash* do objeto de dados) e v é o valor (informações sobre o *peer* servidor que possui o objeto). Ou seja, um objeto de dados, por exemplo, um arquivo, é armazenado no *peer* que possui aquele objeto, mas o valor do *hash* do objeto (chave c) e as informações sobre o *peer* (valor v) são armazenados no sucessor de c na forma do par (c, v). Isso significa que o *peer* que armazena o objeto de dados e o *peer* que detém o par (c, v) não são necessariamente os mesmos.

Tabela de derivação

Um nó no algoritmo Chord deve ser capaz de resolver uma consulta: dada uma chave, o nó deve ser capaz de encontrar o identificador do nó responsável por aquela chave ou encaminhar a consulta para outro nó. O encaminhamento, contudo, requer que cada nó possua uma tabela de roteamento. O Chord exige que cada nó conheça *m* nós sucessores e um predecessor. Cada nó cria uma tabela de roteamento, chamada no Chord de tabela de derivação (*finger table*), que se parece com a Tabela 2.14. Note que a chave-alvo na linha *i* é $N + 2^{i-1}$.

A Figura 2.49 mostra apenas a coluna de sucessores para um anel com alguns nós e chaves. Perceba que a primeira linha ($i = 1$) efetivamente fornece o sucessor do nó. Também adicionamos o ID do nó predecessor, algo necessário, como veremos mais adiante.

Tabela 2.14 Tabela de derivação (*finger table*).

i	Chave-alvo	Sucessor da chave-alvo	Informação sobre o sucessor
1	N + 1	Sucessor de N + 1	Endereço IP e porta do sucessor
2	N + 2	Sucessor de N + 2	Endereço IP e porta do sucessor
...	⋮	⋮	⋮
m	$N + 2^{m-1}$	Sucessor de $N + 2^{m-1}$	Endereço IP e porta do sucessor

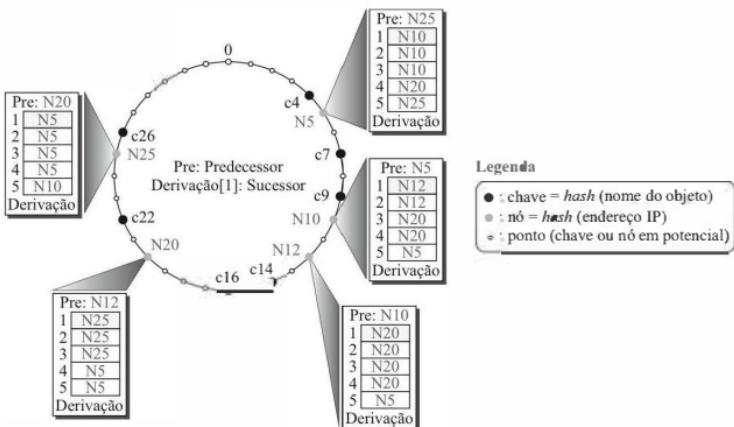



Figura 2.49 Exemplo de um anel no Chord.

Interface

Para funcionar, o Chord precisa de um conjunto de operações conhecidas como a interface Chord. Nesta seção, discutimos algumas dessas operações para dar uma ideia do que está por trás do protocolo Chord.

Busca

A operação mais usada no Chord é provavelmente a busca (*lookup*). O Chord foi projetado para permitir que seus *peers* compartilhem os serviços disponíveis. Para encontrar o objeto a ser compartilhado, um *peer* precisa saber qual é o nó responsável pelo objeto: o *peer* que armazena uma referência para ele. Mencionamos que, no Chord, um *peer* que é o sucessor de um conjunto de chaves no anel é o *peer* responsável por aquelas chaves. Encontrar o nó responsável corresponde, então, a encontrar o sucessor de uma chave. A Tabela 2.15 mostra o código usado para a operação de *busca*.

A função de busca está escrita usando a abordagem *top-down*. Se o nó é responsável pela chave, ele retorna o seu próprio ID; caso contrário, invoca a função busca_sucessor. A função busca_sucessor invoca a função busca_predecessor. A última função invoca a função busca_predecessor_mais_próximo. A abordagem modular nos permite usar as três funções em outras operações em vez de redefini-las.

Discutiremos mais a fundo a função de busca. Se o nó não é responsável pela chave, a função de busca invoca a função busca_sucessor para encontrar o sucessor do ID que lhe foi passado como parâmetro. A codificação da função busca_sucessor pode ser bastante simples se primeiramente encontrarmos o predecessor da chave. O nó predecessor pode facilmente nos ajudar a encontrar o próximo nó no anel, pois o primeiro valor na tabela de derivação do nó predecessor (derivação[1]) nos dá o ID do nó sucessor. Além disso, ter uma função para encontrar o nó predecessor de uma chave pode ser útil para outras funções que escreveremos mais adiante. Infelizmente, um nó normalmente não consegue encontrar o predecessor de uma chave de forma autônoma; a chave pode estar localizada longe do nó. Como explicamos anteriormente, um nó tem conhecimento limitado sobre os outros; a tabela de derivação contém um máximo de aproximadamente *m* outros nós (há algumas

Tabela 2.15 Busca (*Lookup*).

```
busca (chave)
{
        se (nó é responsável pela chave)
                retorne (ID do nó)
        senão
                retorne busca_sucessor (chave)
}
busca_sucessor (id)
{
        x = busca_ predecessor (id)
        retorne x.derivação[1]
}
busca_predecessor (id)
{
        x = N                                               // N é o nó atual
        enquanto (id ∉ (x, x.derivação[1]]
        {
                x = x.busca_predecessor_mais_próximo (id)   // Deixe o x encontrá-lo
        }
        retorne x
}

busca_predecessor_mais_próximo (id)
{
        para (i = m até 1)
        {
                se (derivação[i] ∈ (N, id))                 // N é o nó atual
                        retorne (derivação [i])
        }
        retorne N                                           // O próprio nó é o predecessor mais próximo
}
```

duplicações na tabela de derivação). Por isso, o nó precisa da ajuda de outros nós para encontrar o predecessor de uma chave. Isso pode ser feito com a função busca_predecessor_mais_próximo como uma Chamada de Procedimento Remoto (RPC – Remote Procedure Call). Uma RPC consiste em invocar uma função (procedimento) a ser executada em um nó remoto e retornar o resultado ao nó que iniciou a RPC. Utilizamos a expressão *x.função* no algoritmo, onde *x* é a identidade do nó remoto e *função* é o procedimento a ser executado. O nó usa essa função para encontrar outro nó que esteja mais próximo do predecessor do que ele mesmo. Então, passa a tarefa de encontrar o nó predecessor para o outro nó. Em outras palavras, se o nó A quer encontrar o nó X, ele encontra o nó B (um predecessor mais próximo) e passa essa tarefa para B. O nó B entra então em ação e tenta encontrar X, ou passa a tarefa para outro nó, C. O processo continua até que o nó que conhece o nó predecessor o encontra.

Exemplo 2.16

Considere que o nó N5 na Figura 2.49 queira encontrar o nó responsável pela chave c14. A Figura 2.50 mostra a sequência de 8 eventos que ocorrem nesse caso. Após o evento 4, no qual a função busca_predecessor_mais_próximo retorna N10, a função busca_predecessor pede a N10 que retorne o valor de derivação[1], que é N12. Nesse momento, N5 descobre que N10 não é o predecessor de c14. O nó N5, então, pede a N10 que encontre o predecessor mais próximo de c14, que retorna como N12 (eventos 5 e 6). O nó N5 agora pede o valor de derivação[1] do nó N12, que retorna como N20. O nó N5, então, verifica e percebe que N12 é de fato o predecessor de c14. Essa informação é passada para a função busca_sucessor (evento 7). N5 agora pede o valor de derivação[1] do nó N12, recebendo como resultado N20. A busca é encerrada e N20 é o sucessor do c14.

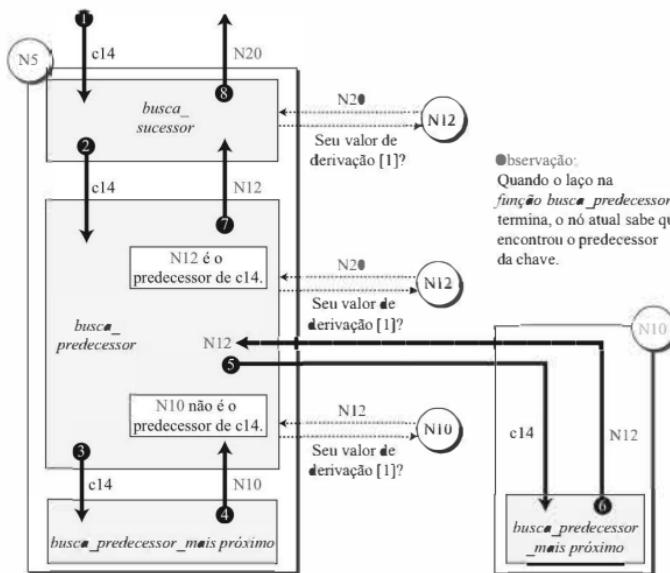



Figura 2.50 Esquema para o Exemplo 2.16.

Estabilizar

Antes de discutirmos como os nós entram e saem de um anel, é preciso enfatizar que qualquer alteração no anel (tal como a entrada e saída de um nó ou de um grupo de nós) pode desestabilizar o anel. Uma das operações definidas no Chord é chamada *estabilizar*. Cada nó no anel periodicamente utiliza essa operação para validar as informações sobre o seu sucessor e para permitir que ele valide suas informações sobre o predecessor. O nó N usa o valor de derivação[1], S, para pedir ao nó S que retorne seu predecessor, P. Se o valor retornado por essa consulta, P, estiver entre N e S, significa que existe um nó cujo ID é igual a P e que se encontra entre N e S. Então, o nó N faz de P o seu sucessor e notifica P para que o nó N se torne seu antecessor. A Tabela 2.16 ilustra a operação de estabilizar.

Tabela 2.16 Estabilizar.

```
estabilizar()
{
        P = derivação[1].Pre                            // Pede que o sucessor retorne seu predecessor
        se (P ∈ (N, derivação[1])) derivação[1] = P      // P é o possível sucessor de N
        derivação[1].notificar (N)                       // Notifica P para que ele troque seu predecessor
}
notificar(x)
{
        se (Pre = null ou x ∈ (Pre, N)) Pre = x
}
```

Restaurar_Tabela_Derivação

Desestabilizações do anel podem alterar a tabela de derivação de até *m* nós. Outra operação definida no Chord é a chamada *restaurar_tabela_derivação*. Cada nó do anel deve invocar periodicamente essa função para manter a tabela de derivação atualizada. Para evitar elevação no tráfego do sistema, cada nó deve atualizar apenas uma das entradas da tabela em cada chamada. A entrada é escolhida aleatoriamente. A Tabela 2.17 mostra o código para a operação.

Tabela 2.17 Restaurar_Tabela_Derivação.

```
restaurar_tabela_derivação()
{
        Gerar (i ∈ [1, m])                              // Gera i aleatoriamente de tal forma que 1 < i ≤ m
        derivação[i] = busca_sucessor (N + 2^(i-1))      // Encontra o valor de derivação[i]
}
```

Associação

Quando um *peer* entra no anel, ele usa a operação *associar* (*join*) e o ID de outro *peer* conhecido para buscar seu sucessor, definindo o predecessor como *null* (ou seja, vazio). Ele imediatamente invoca a função *estabilizar* para validar seu sucessor. O nó, então, pede ao seu sucessor que invoque a função *mover-chave*, que transfere as chaves pelas quais o novo *peer* será responsável. A Tabela 2.18 mostra o código para essa operação.

Tabela 2.18 Associar.

```
Associar (x)
{
        inicializar (x)
        derivação[1].mover_chaves (N)
}
Inicializar (x)
{
        Pre = null
        se (x = null) derivação[1] = N
```

(*Continua*)

Tabela 2.18 Associar (*Continuação*)

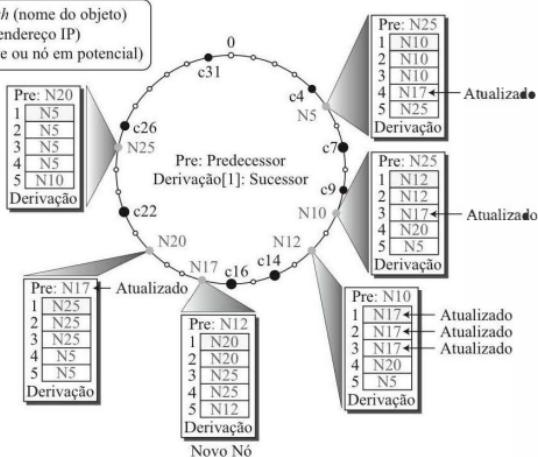
```
            senão derivação[1] = x.busca_sucessor (N)
}


Mover_chaves (x)
{
            para (cada chave c)
{
                  se (x ∈ [k, N)) move (k para nó x)          // N é o nó atual
}
}
```

É óbvio que, após essa operação, a tabela de derivação do nó que associou-se estará vazia e as tabelas de derivação de até *m* predecessores estarão desatualizadas. As operações de *estabilizar* e *restaurar_tabela_derivação* que são executadas periodicamente após esse evento gradualmente estabilizarão o sistema.

Exemplo 2.17

Considere que o nó N17 associa-se ao anel na Figura 2.49 com a ajuda de N5. A Figura 2.51 mostra o anel após ele ser estabilizado. O processo é mostrado a seguir:

1. N17 usa o algoritmo *Inicializar* (5) para definir seu predecessor como *null* e seu sucessor (derivação[1]) como N20.
2. N17 pede, então, que N20 envie c14 e c16 para N17, pois N17 é agora responsável por essas chaves.
3. Após um intervalo limite, N17 usa a operação *estabilizar* para validar seu próprio sucessor (que é N20) e pede a N20 que mude seu predecessor para N17 (usando a função *notificar*).
4. Quando N12 usa a função *estabilizar*, o predecessor de N17 é atualizado para N12.
5. Finalmente, quando alguns nós usam a função *restaurar_tabela_derivação*, as tabelas de derivação dos nós N17, N10, N5 e N12 são alteradas.

Saída ou falha

Se um *peer* deixa o anel ou se o *peer* (não o anel) falhar, a operação do anel será interrompida a menos que o anel se estabilize. Cada nó troca mensagens de *ping* e *pong* com os seus vizinhos para descobrir se eles estão ativos. Quando um nó não recebe uma mensagem de *pong* em resposta à sua mensagem de *ping*, sabe que o seu vizinho está inativo.

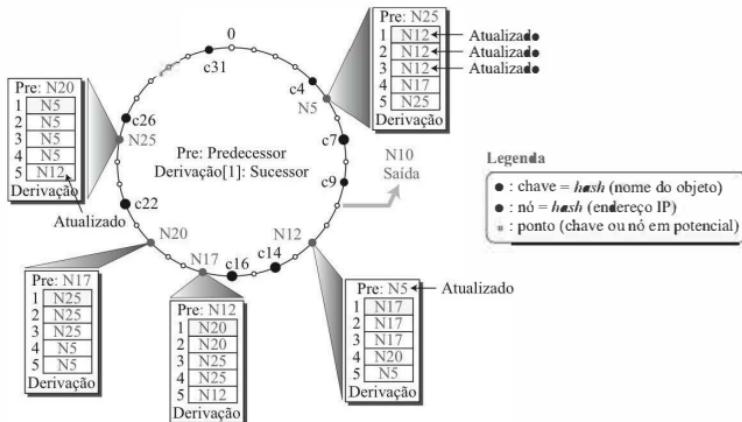
Embora o uso das operações *estabilizar* e *restaurar_tabela_derivação* possam restaurar o anel depois de uma saída ou falha, o nó que detecta o problema pode imediatamente invocar essas operações sem esperar pelo tempo limite para fazê-lo. Um problema importante é que as funções *estabilizar* e *restaurar_tabela_derivação* podem não funcionar corretamente se vários nós deixarem o anel ou falharem ao mesmo tempo. Por essa razão, o Chord exige que cada nó monitore *r* sucessores (o valor de *r* depende da implementação). O nó pode sempre ir para o próximo sucessor se os anteriores não estiverem disponíveis.

Outro problema nesse caso é que os dados gerenciados pelo nó que deixou o anel ou falhou já não estarão disponíveis. O Chord estipula que apenas um nó deve ser responsável por um conjunto de dados e referências, mas o Chord também exige que os dados e referências sejam duplicados em outros nós, nesse caso.

Figura 2.51 Esquema para o Exemplo 2.17.

Exemplo 2.18

Consideremos que um nó, N10, deixa o anel da Figura 2.51. A Figura 2.52 mostra o anel depois de ele ter sido estabilizado.

Figura 2.52 Esquema para o Exemplo 2.18.

O processo é mostrado a seguir:

1. O nó N5 percebe a saída de N10 quando não recebe uma mensagem de *pong* em resposta a sua mensagem de *ping*. O nó N5 muda seu sucessor (o valor de derivação [1]) para N12 (o segundo na lista de sucessores).
2. O nó N5 ativa imediatamente a função *estabilizar* e pede a N12 que mude seu predecessor para N5.
3. Com sorte, as chaves c7 e c9, que estavam sob a responsabilidade de N10, foram duplicadas em N12 antes da saída de N10.
4. Depois de algumas invocações à função *restaurar_tabela_derivação*, os nós N5 e N25 atualizam suas tabelas de derivação, conforme mostrado na figura.

Aplicações

O Chord é usado em diversas aplicações, incluindo o Sistema de Arquivos Colaborativo (CFS – Collaborative File System), Con-Chord e o Sistema de Nomes de Domínio Distributivo (DDNS – Distributive Domain Name System).

2.4.4 Pastry

Outro protocolo popular que segue o paradigma P2P é o **Pastry**, concebido por Rowstron e Druschel. O Pastry utiliza uma DHT, como descrito anteriormente, mas existem algumas diferenças fundamentais entre o Pastry e o Chord no que diz respeito ao espaço de identificadores e ao processo de roteamento, que descreveremos a seguir.

Espaço de identificadores

No Pastry, assim como no Chord, os nós e os dados são representados por identificadores de *m bits*, criando um espaço de identificadores com 2^m de pontos distribuídos uniformemente em um círculo no sentido horário. Um valor comum para m é 128. O protocolo usa o algoritmo de *hash* SHA-1 com $m = 128$. No entanto, nesse protocolo, um identificador é visto como uma cadeia de n dígitos na base 2^b, onde b é normalmente 4 e $n = (m/b)$. Em outras palavras, um identificador é um número de 32 dígitos na base 16 (hexadecimal). Nesse espaço de identificadores, cada chave é armazenada no nó cujo identificador é numericamente mais próximo da chave, estratégia definitivamente diferente daquela utilizada pelo Chord. No Chord, uma chave é armazenada em seu nó sucessor; no Pastry, uma chave é armazenada no nó que for numericamente mais próximo da chave, podendo ser ele o nó sucessor ou predecessor.

Roteamento

Um nó no Pastry deve ser capaz de resolver uma consulta; dada uma chave, o nó deve encontrar o identificador do nó responsável por ela ou encaminhar a consulta para outro nó. Cada nó no Pastry usa duas entidades para fazer isso: uma *tabela de roteamento* e um *conjunto de folhas*.

Tabela de roteamento

O Pastry exige que cada nó mantenha uma tabela de roteamento com n linhas e (2^b) colunas. Em geral, quando $m = 128$ e $b = 4$, temos 32 (128/4) linhas e 16 colunas ($2^{128} = 16^{32}$). Ou seja, temos uma linha para cada dígito do identificador e uma coluna para cada valor hexadecimal (0 a F). A Tabela 2.19 mostra a estrutura básica da tabela de roteamento para o caso geral. Na tabela para o nó N, a célula na linha i e coluna j (ou seja, o valor de Tabela $[i, j]$) fornece o identificador de um nó

(caso ele exista) cujos i dígitos mais à esquerda coincidem com aqueles do identificador de N e cujo dígito na posição $(i + 1)$ vale j. A primeira linha, 0, mostra a lista de nós ativos cujos identificadores não têm um prefixo em comum com N. A linha 1 mostra uma lista de nós ativos de mesmo dígito mais à esquerda que o identificador do nó N. De maneira similar, a linha 31 mostra a lista de todos os nós ativos que compartilham os 31 dígitos mais à esquerda com o nó N, tendo apenas o último dígito diferente.

Tabela 2.19 Tabela de roteamento para um nó no Pastry.

Tamanho do prefixo em comum	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0																
1																
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
31																

Por exemplo, se $N = (574A234B12E374A2001B23451EEE4BCD)_{16}$, então o valor de Tabela [2, D] pode ser o identificador de um nó tal como (**57D...**). Perceba que os dois dígitos mais à esquerda são 57, que coincidem com os dois primeiros dígitos de N, mas o próximo dígito é D, o valor correspondente à D-ésima coluna. Se houver mais nós com o prefixo 57D, o mais próximo, de acordo com a ***métrica de proximidade*** definida, é o escolhido, e seu identificador é inserido nesta célula. A métrica de proximidade é uma medida de distância determinada pela aplicação que usa a rede, e pode ser baseada no número de saltos entre os dois nós, no RTT entre os dois nós ou em outras métricas.

Conjunto de folhas

Outra entidade utilizada no roteamento é um conjunto de 2^b identificadores (o tamanho de uma linha na tabela de roteamento) chamado *conjunto de folhas* (*leaf set*). Metade do conjunto consiste em uma lista de identificadores numericamente menores do que o identificador do nó atual; a outra metade é uma lista de identificadores numericamente maiores do que o identificador do nó atual. Ou seja, o conjunto de folhas fornece os identificadores de 2^{b-1} nós ativos que antecedem o nó atual no anel e a lista de 2^{b-1} nós que vêm depois do nó atual no anel.

Exemplo 2.19

Consideremos que $m = 8$ *bits* e $b = 2$. Isto significa que pode haver até $2^m = 256$ identificadores, e cada identificador tem $m/b = 4$ dígitos na base $2^b = 4$. A Figura 2.53 mostra a situação em que há alguns nós ativos e algumas chaves mapeadas para esses nós. A chave c1213 é armazenada em dois nós porque é equidistante em relação à deles. Isto fornece alguma redundância que pode ser usada se um dos nós falhar. A figura também mostra as tabelas de roteamento e os conjuntos de folhas para quatro nós selecionados utilizados nos exemplos descritos a seguir. Na tabela de roteamento para o nó N0302, por exemplo, escolhemos o nó 1302 para ser inserido em Tabela [0,1] porque consideramos que ele está mais próximo de N0302 de acordo com a métrica de proximidade empregada. Usamos a mesma estratégia para as outras entradas. Observe que uma célula em cada linha de cada tabela foi sombreada porque corresponde ao dígito do identificador do nó e, portanto, nenhum identificador de nó pode ser ali inserido. Existem também algumas células vazias porque não existem nós ativos na rede no momento que satisfaçam os requisitos para preenchê-la; quando alguns novos nós entrarem na rede, eles poderão ser inseridos nestas células.

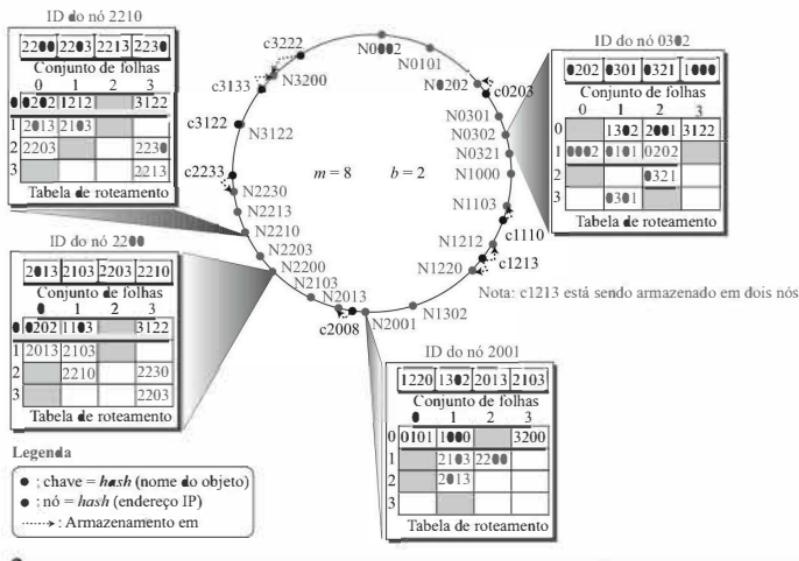



Figura 2.53 Exemplo de anel do Pastry.

Busca

Como discutido para o Chord, uma das operações utilizadas no Pastry é a busca (*lookup*): dada uma chave, é preciso encontrar o nó que armazena as informações sobre ela ou a própria chave. A Tabela 2.20 descreve a operação de busca em pseudocódigo. Nesse algoritmo, N é o identificador do nó local, o nó que recebe uma mensagem e precisa encontrar o nó que armazena a chave definida na mensagem.

Exemplo 2.20

Na Figura 2.53, consideramos que o nó N2210 recebe uma consulta em busca do nó responsável pela chave 2008. Como ele não é responsável pela chave, ele primeiro verifica o seu conjunto de folhas. A chave 2008 não se encontra entre os valores em seu conjunto de folhas, de modo que o nó precisa utilizar a sua tabela de roteamento. Uma vez que o comprimento do prefixo em comum é 1, p = 1. O valor do dígito na posição 1 da chave é v = 0. O nó verifica o identificador em Tabela [1, 0], que é de 2013. A consulta é encaminhada para o nó 2013, que é de fato responsável pela chave. Ele envia sua informação para o nó que efetuou a consulta.

Exemplo 2.21

Na Figura 2.53, consideramos que o nó N0302 recebe uma consulta em busca do nó responsável pela chave 0203. Ele não é responsável pela chave, mas ela está em seu conjunto de folhas. O nó mais próximo nesse conjunto é o nó N0202. A consulta é enviada para esse nó, que é de fato responsável pela chave. O nó N0202 envia sua informação para o nó que efetuou a consulta.

Tabela 2.20 Busca (*Lookup*).

```
Busca (chave)
{
        se (chave está entre o conjunto de folhas de N)
                encaminhe a mensagem para o nó mais próximo no conjunto de folhas
        senão
                rotear (chave, Tabela)
}
rotear (chave, Tabela)
{
        p = tamanho do prefixo compartilhado entre chave e N
        v = valor do digito na posição p da chave                // Posição começa em 0
        se (Tabela [p, v] existe)
                encaminhe a mensagem para o nó em Tabela [p, v]
        senão
                encaminhe a mensagem para um nó que compartilha um prefixo tão longo quando o nó atual, mas
                numericamente mais próximo da chave
}
```

Associação

O processo de entrada no anel do Pastry é mais simples e mais rápido do que no Chord. O novo nó, X, deve conhecer pelo menos um nó **N0**, que deve estar próximo de X (baseado na métrica de proximidade usada); isso pode ser feito por meio da execução de um algoritmo denominado Descoberta de Nó Vizinho (Nearby Node Discovery). O nó X envia uma *mensagem de associação* (*join*) para **N0**. Em nossa discussão, consideraremos que o identificador **N0** não tem um prefixo em comum com o identificador X. Os passos a seguir mostram como o nó X cria sua tabela de roteamento e conjunto de folhas:

1. O nó N0 envia o conteúdo de sua linha 0 para o nó X. Como os dois nós não têm um prefixo comum, o nó X usa as partes apropriadas dessa informação para construir sua linha 0. O nó N0, então, trata a mensagem de associação como uma mensagem de busca, assumindo que o identificador X é uma chave. Ele encaminha a mensagem de associação a um nó, N1, cujo identificador é o mais próximo de X.
2. O nó N1 envia o conteúdo de sua linha 1 para o nó X. Como os dois nós têm um prefixo em comum, o nó X usa as partes apropriadas da informação para construir a sua linha 1. O nó N1, então, trata a mensagem de associação como uma mensagem de busca, assumindo que o identificador X é uma chave. Ele encaminha a mensagem de associação a um nó, N2, cujo identificador é o mais próximo de X.
3. O processo continua até que a tabela de roteamento do nó X esteja completa.
4. O último nó do processo, que tem o mais longo prefixo em comum com X, também envia seu conjunto de folhas para o nó X, o qual passa a ser o conjunto de folhas de X.
5. O nó X, então, troca informações com nós em sua tabela de roteamento e o conjunto de folhas para aprimorar a própria informação de roteamento e também para permitir que aqueles nós atualizem suas informações.

Exemplo 2.22

A Figura 2.54 mostra como um novo nó X cujo identificador é N2212 utiliza as informações de quatro nós da Figura 2.53 para criar a sua tabela de roteamento e conjunto de folhas iniciais, assim que ele se associa ao anel. Perceba que o conteúdo dessas duas tabelas se aproximará do esperado ao longo do processo de atualização. Nesse exemplo, consideramos que o nó 0302 seja um nó próximo ao nó 2212 com base na métrica de proximidade utilizada.

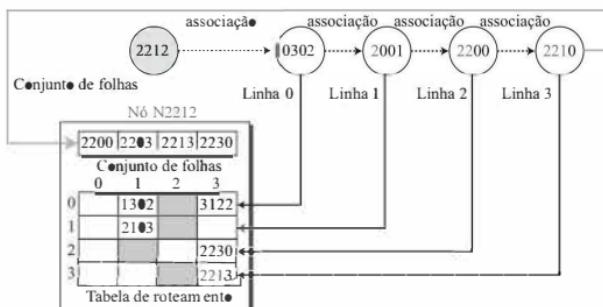



Figura 2.54 Esquema para o Exemplo 2.22.

Saída ou falha

Cada nó do Pastry periodicamente testa se os nós em seu conjunto de folhas e em sua tabela de roteamento continuam ativos por meio de mensagens de sondagem (*probe messages*). Se um nó local descobre que um nó em seu conjunto de folhas não está respondendo às mensagens de sondagem, ele considera que o nó falhou ou deixou o anel. Para substituí-lo em seu conjunto de folhas, o nó local contacta o nó ativo de seu conjunto de folhas que tenha o identificador mais elevado, reparando seu conjunto de folhas com as informações provenientes daquele nó. Como existe uma sobreposição no conjunto de folhas de nós próximos, esse processo é bem-sucedido.

Se um nó local descobre que outro em sua tabela de roteamento, Tabela [i, j], não responde às mensagens da sondagem, ele envia uma mensagem a um nó ativo na mesma linha e pede o identificador na posição Tabela [i, j] dele. Esse identificador substitui o nó que falhou ou deixou o anel.

Aplicação

O Partry é usado em algumas aplicações, incluindo PAST, um sistema de arquivos distribuído, e SCRIBE, um sistema descentralizado de publicação/assinatura.

2.4.5 Kademlia

Outra rede *peer-to-peer* baseada em DHT é o Kademlia, concebido por Maymounkov e Mazières. O Kademlia, assim como o Pastry, roteia mensagens de acordo com a distância entre os nós, mas a interpretação da métrica de distância no Kademlia é diferente da adotada no Pastry, como descreveremos a seguir. Na rede Kademlia, a distância entre dois identificadores (sejam eles nós ou chaves) é medida usando a operação de ou-exclusivo (XOR) *bit* a *bit* entre tais identificadores. Em outras palavras, se X e Y são dois identificadores, temos

$$\text{distância}(x, y) = x \oplus y$$

A métrica baseada em XOR tem quatro propriedades que esperamos obter quando usamos distâncias geométricas entre dois pontos, mostradas a seguir:

$x \oplus x = 0$	A distância de um nó a si mesmo é zero.
$x \oplus y > 0$ se $x \neq y$	A distância entre quaisquer dois nós distintos é maior do que zero.
$x \oplus y = y \oplus x$	A distância entre x e y é igual à distância entre y e x.
$x \oplus z < x \oplus y + y \oplus z$	A relação triangular é satisfeita.

Espaço de identificadores

No Kademlia, nós e dados são representados usando identificadores de *m bits* que criam um espaço de identificadores contendo 2^m pontos distribuídos nas folhas de uma árvore binária. O protocolo usa o algoritmo de *hash* SHA-1 com $m = 160$.

Exemplo 2.23

Por razões de simplicidade, consideremos que $m = 4$. Neste espaço, podemos ter 16 identificadores distribuídos pelas folhas de uma árvore binária. A Figura 2.55 mostra esse caso com apenas oito nós ativos e cinco chaves.

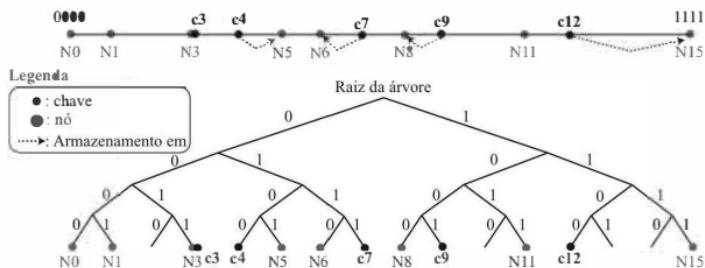



Figura 2.55 Esquema para o Exemplo 2.23.

Conforme mostrado na figura, a chave c3 é armazenada em N3 porque $3 \oplus 3 = 0$. Embora a chave c7 pareça numericamente equidistante de N6 e N8, ela é armazenada apenas em N6 porque $6 \oplus 7 = 1$, enquanto $6 \oplus 8 = 14$. Outro ponto interessante é que a chave c12 é numericamente mais próxima de N11, mas é armazenada em N15 porque $11 \oplus 12 = 7$, enquanto $15 \oplus 12 = 3$.

Tabela de roteamento

O Kademlia mantém apenas uma tabela de roteamento para cada nó; não existe um conjunto de folhas. Cada nó na rede divide a árvore binária em m subárvores que não incluem o próprio nó. A subárvore i inclui os nós que compartilham os i *bits* mais à esquerda (prefixo em comum) com o nó correspondente. A tabela de roteamento é composta por m linhas, mas apenas uma coluna. Durante nossa discussão, consideraremos que cada linha contém o identificador de um dos nós na subárvore correspondente, mas depois mostramos que o Kademlia permite até k nós em cada linha. A ideia é a mesma utilizada no Pastry, mas o comprimento do prefixo em comum baseia-se no número de *bits* em vez do número de dígitos na base 2^b. A Tabela 2.21 mostra a tabela de roteamento.

Tabela 2.21 Tabela de roteamento para um nó no Kademlia.

Tamanho do prefixo em comum	Identificação
0	Nó(s) mais próximo(s) na subárvore com prefixo em comum de comprimento 0
1	Nó(s) mais próximo(s) na subárvore com prefixo em comum de comprimento 1
⋮	⋮
$m-1$	Nó(s) mais próximo(s) na subárvore com prefixo em comum de comprimento $m-1$

Exemplo 2.24

Encontraremos a tabela de roteamento para o Exemplo 2.23. Para simplificar o exemplo, consideramos que cada linha utiliza apenas um identificador. Como m = 4, cada nó possui quatro subárvores correspondentes às quatro linhas na tabela de roteamento. O identificador em cada uma representa o nó que está mais próximo do nó atual na subárvore correspondente. A Figura 2.56 mostra todas as tabelas de roteamento, mas apenas três das subárvores. Escolhemos apenas três, de um total de oito, para reduzir o tamanho da figura.

Explicaremos como foi feita a tabela de roteamento, por exemplo, para o nó 6, usando as subárvores correspondentes. As explicações para os outros nós são equivalentes.

a. Na linha 0, precisamos inserir o identificador do nó mais próximo na subárvore com prefixo em comum de comprimento $p = 0$. Há três nós nessa subárvore (N8, N11 e N15), mas N15 é o mais próximo de N6 porque N6 ⊕ N8 = 14, N6 ⊕ N11 = 13 e N6 ⊕ N15 = 9. N15 é inserido na linha 0.

b. Na linha 1, precisamos inserir o identificador do nó mais próximo na subárvore com prefixo em comum de comprimento $p = 1$. Há três nós nessa subárvore (N0, N1 e N3), mas N3 é o mais próximo de N6 porque N6 ⊕ N0 = 6, N6 ⊕ N1 = 7 e N6 ⊕ N3 = 5. N3 é inserido na linha 1.

c. Na linha 2, precisamos inserir o identificador do nó mais próximo na subárvore com prefixo em comum de comprimento $p = 2$. Há apenas um nó nessa subárvore (N5), que é aqui inserido.

d. Na linha 3, precisamos inserir o identificador do nó mais próximo na subárvore com prefixo em comum de comprimento $p = 3$. Não há nós nessa subárvore, então a linha fica vazia.

Exemplo 2.25

Na Figura 2.56, consideramos que o nó N0 $(0000)_2$ recebe uma mensagem de busca tendo como alvo o nó responsável por c12 $(1100)_2$. O comprimento do prefixo em comum entre os dois identificadores é 0. O nó N0 encaminha a mensagem para o nó na linha 0 de sua tabela de roteamento, o nó N8. Agora, o nó N8 $(1000)_2$ precisa procurar o nó mais próximo de c12 $(1100)_2$. O comprimento do prefixo em comum entre os dois identificadores é 1. O nó N8 encaminha a mensagem para o nó na linha 1 de sua tabela de roteamento, o N15, o qual é responsável por c12. O processo de roteamento se encerra. A rota é N0 → N8 → N15. É interessante notar que o nó N15, $(1111)_2$, e c12, $(1100)_2$, têm um prefixo em comum de comprimento 2, mas a linha 2 de N15 está vazia, o que significa que o próprio N15 é responsável por c12.

Exemplo 2.26

Na Figura 2.56, consideramos que o nó N5 $(0101)_2$ recebe uma mensagem de busca tendo como alvo o nó responsável por c7 $(0111)_2$. O comprimento do prefixo em comum entre os dois identificadores é 2. O N5 envia a mensagem para o nó na linha 2 de sua tabela de encaminhamento, o N6, que é o responsável por c7. O processo de roteamento se encerra. A rota é N5 → N6.

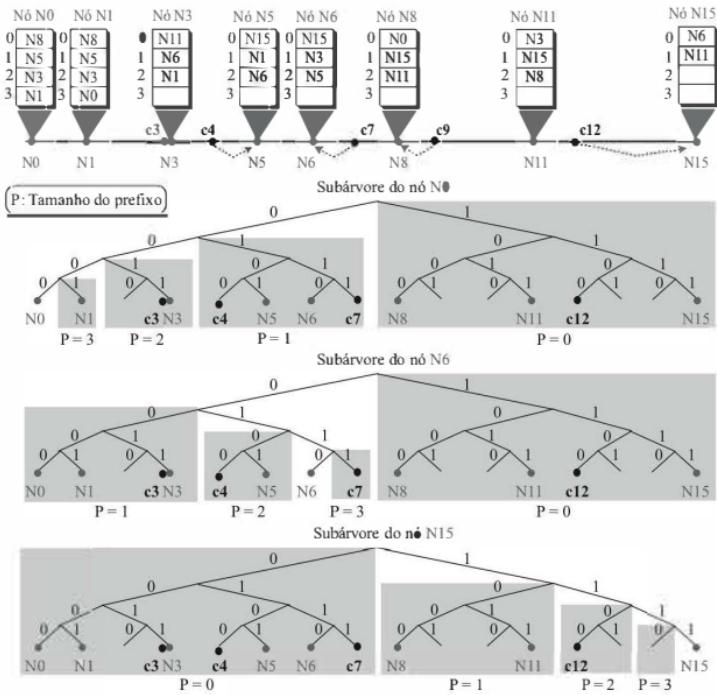



Figura 2.56 Esquema para o Exemplo 2.24.

Exemplo 2.27

Na Figura 2.56, consideramos que o nó N11 $(1011)_2$ recebe uma mensagem de busca tendo como alvo o nó responsável por c4 $(0100)_2$. O comprimento do prefixo em comum entre os dois identificadores é 0. O nó N11 envia a mensagem para o nó na linha 0 de sua tabela de roteamento, o N3. Agora, o nó N3 $(0011)_2$ precisa procurar pelo nó mais próximo de c4 $(0100)_2$. O comprimento do prefixo em comum entre os dois identificadores é 1. O nó N3 envia a mensagem para o nó na linha 1 de sua tabela de roteamento, o N6. Agora, o nó N6 $(0110)_2$ precisa procurar o nó mais próximo de c4 $(0100)_2$. O comprimento do prefixo em comum entre os dois identificadores é 2. O nó N6 envia a mensagem para o nó na linha 2 de sua tabela de roteamento, o N5, responsável por c4. O processo de roteamento se encerra. A rota é N11 → N3 → N6 → N5.

K-Buckets

Em nossa discussão anterior, consideramos que cada linha na tabela de roteamento contém apenas um nó na subárvore correspondente. Para ser mais eficiente, o Kademlia exige que cada linha tenha capacidade para até *k* nós da subárvore correspondente. O valor de *k* é independente do sistema, mas para uma rede real, recomenda-se um valor de *k* por volta de 20. Por isso, cada linha da tabela de roteamento se comporta como se fosse um balde (*bucket*, em inglês), e a linha é então chamada

k-bucket. Ter mais do que um nó em cada linha permite que o nó tenha alternativas quando outro deixa a rede ou falha. O Kademlia mantém em um balde os nós que ficaram conectados na rede por um longo tempo. Prova-se que os nós que estão conectados há um longo tempo provavelmente permanecerão conectados por um longo tempo.

Consulta em paralelo

Como existem vários nós em um *k-bucket*, o Kademlia permite o envio de α consultas em paralelo para α nós no topo do *k-bucket*. Isto reduz o atraso se um nó falhar e não puder responder à consulta.

Atualização simultânea

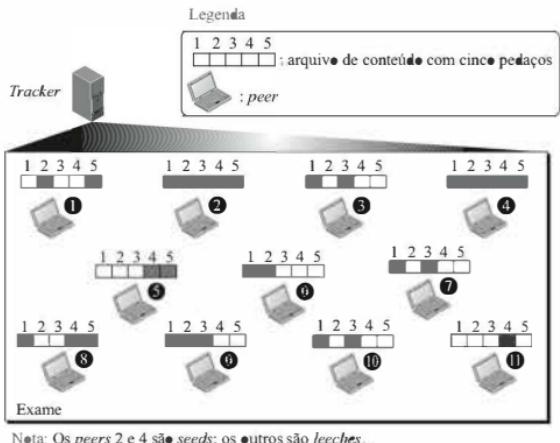
Outra característica interessante do Kademlia é a atualização simultânea. Sempre que um nó recebe uma consulta ou uma resposta, ele atualiza seu *k-bucket*. Se várias consultas enviadas a um nó não apresentam resposta, o nó que enviou a consulta pode remover o nó destino do *k-bucket* correspondente.

Associação

Assim como no Pastry, um nó que deseja entrar na rede precisa conhecer pelo menos um outro nó. O nó ingressante envia seu identificador para o nó como se fosse uma chave a ser encontrada. A resposta recebida permite que o novo nó crie seus *k-buckets*.

Saída ou falha

Quando um nó deixa a rede ou falha, outros nós atualizam seus *k-buckets* usando o processo simultâneo descrito anteriormente.

2.4.6 Uma rede P2P popular: o BitTorrent

O BitTorrent é um protocolo P2P projetado por Bram Cohen para compartilhar arquivos grandes entre grupos de *peers*. No entanto, o termo *compartilhar* nesse contexto é diferente daquele usado em outros protocolos de compartilhamento de arquivos. Em vez de um *peer* permitir que outro *peer* faça o *download* do arquivo completo, um grupo de *peers* participa do processo para propiciar a todos os *peers* do grupo uma cópia do arquivo. O compartilhamento de arquivos é feito por meio de um processo colaborativo chamado *torrent*. Cada *peer* participando em um *torrent* obtém pedaços do arquivo grande de outro *peer* que tenha tal arquivo e, ao mesmo tempo, fornece pedaços do arquivo para outros *peers* que não o possuem em uma espécie de *tit-for-tat*, um jogo de negociação jogado por crianças*. O conjunto de todos os *peers* que participam em um *torrent* é denominado *swarm*, ou *enxame*. Um *peer* com arquivo de conteúdo completo de um enxame é chamado *seed*, ou *semente*; um *peer* que tem apenas parte do arquivo e quer obter o restante do mesmo é chamado *leech*, ou *sanguessuga*. Em outras palavras, um enxame é uma combinação de *seeds* e *leeches*. O BitTorrent passou por várias versões e apresenta muitas implementações. Em primeiro lugar, descreveremos a versão original, que utiliza um nó central chamado *tracker*, ou *rastreador*. Em seguida, mostraremos como algumas novas versões eliminam o *tracker* por meio do uso de uma DHT.

BitTorrent com um *tracker*

O BitTorrent original apresenta outra entidade em um torrent, chamada *tracker* (ou rastreador), que, como o nome sugere, monitora a operação enxame, conforme descrito mais adiante. A Figura 2.57 mostra um exemplo de um *torrent* com *seeds*, *leeches* e o *tracker*.

Na figura, o arquivo a ser compartilhado, o arquivo de conteúdo, é dividido em cinco partes (conhecidas como pedaços, ou *chunks*). Os *peers* 2 e 4 já têm todos os pedaços, enquanto os

* N. de T.: O *tit-for-tat* refere-se a negociações no estilo "olho por olho", ou "é dando que se recebe".

Nota: Os *peers* 2 e 4 são *seeds*; os outros são *leeches*.

Figura 2.57 Exemplo de um *torrent*.

outros têm apenas alguns deles. Os pedaços que cada *peer* já possui estão sombreados. O *upload* e o *download* de pedaços são contínuos. Alguns *peers* podem deixar o *torrent*, enquanto novos *peers* podem passar a participar dele.

Agora, considere que um novo *peer* quer obter esse mesmo arquivo de conteúdo. O novo *peer* acessa o servidor BitTorrent e fornece o nome do arquivo de conteúdo; recebe um metarquivo com o nome do arquivo de *torrent*, o qual contém as informações sobre os pedaços do arquivo de conteúdo e o endereço do *tracker* que gerencia esse *torrent* em específico. O novo *peer*, então, acessa o *tracker* e recebe os endereços de alguns *peers* no *torrent*, normalmente chamados *vizinhos*. O novo *peer* é agora parte do *torrent* e pode obter e enviar partes do arquivo de conteúdo. Quando tiver todos os pedaços, ele pode abandonar o *torrent* ou permanecer nele com o intuito de ajudar outros *peers*, incluindo os novos que entraram no enxame depois dele, a obter todos os pedaços do arquivo de conteúdo. Nada impede que um *peer* abandone o *torrent* antes de obter todos pedaços e depois volte a participar do enxame mais tarde ou de não voltar novamente.

Embora os processos de entrar, compartilhar e deixar um *torrent* possam parecer simples, o protocolo BitTorrent aplica uma série de políticas com o objetivo de garantir equidade, para incentivar os *peers* a trocar pedaços com outros *peers*, evitar a sobrecarga de um *peer* com os pedidos dos outros e permitir que um *peer* encontre outros que forneçam melhores serviços.

Para evitar a sobrecarga de um *peer* e garantir equidade na rede, cada *peer* precisa limitar o número de conexões simultâneas com seus vizinhos; o valor tipicamente adotado é quatro. Um *peer* marca um vizinho como *unchoked* (desbloqueado) ou *choked* (bloqueado), e também os classifica como interessados ou desinteressados.

Em outras palavras, um *peer* divide suas listas de vizinhos em dois grupos distintos: *unchoked* e *choked*. Também os divide nos grupos *interessado* e *desinteressado*. O grupo *unchoked* refere-se à lista dos *peers* aos quais o *peer* atual está simultaneamente conectado; ele envia e recebe pedaços desse grupo sem parar. O grupo *choked* refere-se à lista de vizinhos aos quais o *peer* não está atualmente conectado, mas aos quais ele pode se conectar no futuro.

A cada 10 segundos, o *peer* atual conecta-se a um *peer* do grupo interessado que esteja *choked*, na esperança de conseguir uma melhor taxa de transferência de dados. Se o novo *peer* fornece uma

taxa melhor do que qualquer um dos *peers* que estejam atualmente no grupo *unchoked*, o novo *peer* pode ser desbloqueado e o *peer* com a menor taxa de transferência de dados no grupo *unchoked* pode ser bloqueado, passando, então, para o grupo *choked*. Assim, os *peers* no grupo *unchoked* sempre têm a maior taxa de transferência de dados dentre os *peers* testados. A aplicação dessa estratégia divide os vizinhos em subgrupos, de modo que os vizinhos com taxas de transferência de dados compatíveis comuniquem-se uns com os outros. A ideia da estratégia de negociação *tit-for-tat* descrita anteriormente pode ser observada nessa política, que segue o estilo "é dando que se recebe".

Para permitir que um *peer* recém-chegado, sem pedaços para compartilhar, também seja capaz de receber pedaços de outros, a cada 30 segundos um *peer* aleatoriamente desbloqueia um único *peer* do grupo interessado (ou seja, promove-o do grupo *choked* para o grupo *unchoked*) independentemente de sua taxa de *upload*. Essa ação é chamada *optimistic unchoking*, ou *desbloqueio otimista*.

O protocolo BitTorrent tenta proporcionar um equilíbrio entre o número de pedaços que cada *peer* pode ter a cada instante usando uma estratégia chamada *rarest-first* (*o mais raro primeiro*). Nela, um *peer* tenta primeiro fazer o *download* dos pedaços com o menor número de cópias repetidas dentre seus vizinhos; assim, os pedaços são distribuídos mais rapidamente.

BitTorrent sem *tracker*

No projeto original do BitTorrent, se o *tracker* falhar, novos *peers* não podem se conectar à rede e o processo de atualização dos nós é interrompido. Existem várias implementações do BitTorrent que eliminam a necessidade de um *tracker* centralizado. Na implementação que descrevemos aqui, o protocolo ainda usa o *tracker*, mas ele não é centralizado. A tarefa de rastreamento é distribuída dentre alguns nós na rede. Nesta seção, mostramos como a DHT do Kademlia pode ser usada para atingir esse objetivo, mas evitamos mostrar em detalhes o protocolo em específico.

No BitTorrent com um *tracker* central, a tarefa do *tracker* é fornecer uma lista de *peers* em um enxame dado um arquivo de metadados que define o *torrent*. Se pensarmos no *hash* dos metadados como sendo a chave e no *hash* da lista de *peers* em um enxame como sendo o valor, podemos deixar alguns nós em uma rede P2P desempenhar o papel de *trackers*. Um novo *peer* que entra no *torrent* envia o *hash* dos metadados (chave) para o nó que ele conhece. A rede P2P usa o protocolo Kademlia para encontrar o nó responsável pela chave. Este envia o *valor*, que na verdade é a lista de *peers* no *torrent* correspondente, para o *peer* entrante. Agora, o *peer* entrante pode usar o protocolo BitTorrent para compartilhar o arquivo de conteúdo com os *peers* presentes na lista recebida.

2.5 PROGRAMAÇÃO USANDO A INTERFACE *SOCKET*

Na Seção 2.2, discutimos os princípios do paradigma cliente-servidor. Na Seção 2.3, discutimos algumas aplicações padronizadas que usam esse paradigma. Nesta seção, mostramos como escrever alguns programas simples do tipo cliente-servidor usando C, uma linguagem de programação procedural. Escolhemos a linguagem C nesta seção por duas razões. Primeiro, tradicionalmente a programação usando *sockets* começou com a linguagem C. Segundo, os recursos de baixo nível característicos da linguagem C revelam melhor algumas sutilezas nesse tipo de programação. No Capítulo 11, expandimos essa ideia usando Java, que permite a criação de um código mais compacto. Não obstante, esta seção pode ser ignorada sem perda de continuidade no estudo do livro.

2.5.1 Interface *socket* em C

Discutimos a interface *socket* na Seção 2.2. Nesta seção, mostramos como essa interface é implementada na linguagem C. A questão mais importante com relação à interface *socket* é compreender o papel de um *socket* na comunicação; ele não tem um *buffer* para armazenar dados a serem enviados ou recebidos, não é capaz de enviar ou receber dados. O *socket* atua simplesmente como uma referência, ou um rótulo. Os *buffers* e variáveis necessários são criados dentro do sistema operacional.

Estrutura de dados do *socket*

A linguagem C define um *socket* como uma estrutura (*struct*). A estrutura *socket* é composta por cinco campos; cada endereço de *socket* em si é uma estrutura composta por cinco campos, conforme mostra a Figura 2.58. Perceba que o programador não deve redefinir essa estrutura, que já está definida nos arquivos de cabeçalho. Analisamos brevemente os cinco campos de uma estrutura de *socket*.

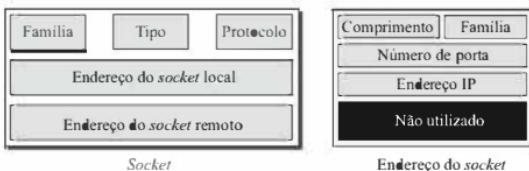



Figura 2.58 Estrutura de dados de um *socket*

- **Família.** Este campo define a família do protocolo (como interpretar os endereços e número de porta). Os valores mais comuns são PF_INET (para a Internet atual, usando IPv4), PF_INET6 (para a nova geração da Internet, usando IPv6) e assim por diante. Usamos PF_INET nesta seção.
- **Tipo.** Este campo define quatro tipos de *sockets*: SOCK_STREAM (para TCP), SOCK_DGRAM (para UDP), SOCK_SEQPACKET (para SCTP) e SOCK_RAW (para aplicações que utilizam diretamente os serviços do IP).
- **Protocolo.** Este campo define o protocolo específico na família. Ele é fixado em 0 para a pilha de protocolos TCP/IP porque esse é o único protocolo na família.
- **Endereço de *socket* local.** Este campo define o *endereço do socket local*. Um endereço de *socket* é em si uma estrutura formada pelos campos *comprimento*, *família* (cujo valor é constante e igual a AF_INET para a pilha de protocolos TCP/IP), *número de porta* (que define o processo), e *endereço IP* (que define o *host* no qual o processo está sendo executado). Ele também contém um campo *não utilizado*.
- **Endereço de *socket* remoto.** Este campo define o *endereço do socket remoto*. A sua estrutura é igual à do endereço de *socket* local.

Arquivos de cabeçalho

Para que seja possível utilizar a definição do *socket* e todas as funções definidas em sua interface, precisamos de um conjunto de arquivos de cabeçalho. Agrupamos todos esses arquivos de cabeçalho em um arquivo chamado *cabecalhos.h*. Esse arquivo deve ser criado no mesmo diretório em que serão colocados os programas e seu nome deve ser incluído em todos eles.

```
// "cabecalhos.h"
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <errno.h>
```

(*Continua*)

(*Continuação*)

```
#include <signal.h>
#include <unistd.h>
#include <string.h>
#include <arpa/innet.h>
#include <sys/wait.h>
```

Comunicação iterativa usando UDP

Como discutimos anteriormente, o UDP fornece um serviço não orientado à conexão, no qual um cliente envia uma solicitação e o servidor envia de volta uma resposta.

Sockets usados no UDP

Na comunicação via UDP, o cliente e o servidor utilizam apenas um *socket* cada. O *socket* criado no lado do servidor nunca é fechado; o criado no lado do cliente é fechado (destruído) quando o processo-cliente é encerrado. A Figura 2.59 mostra o tempo de vida dos *sockets* nos processos servidor e cliente. Em outras palavras, diferentes clientes usam *sockets* distintos, mas o servidor cria um único *socket* e só muda o endereço de *socket* remoto quando um novo cliente o acessa. Isto faz sentido, já que o servidor sabe seu próprio endereço de *socket*, mas não sabe o dos clientes que precisam acessá-lo; ele precisa esperar pelo acesso do cliente para preencher esse campo do *socket*.

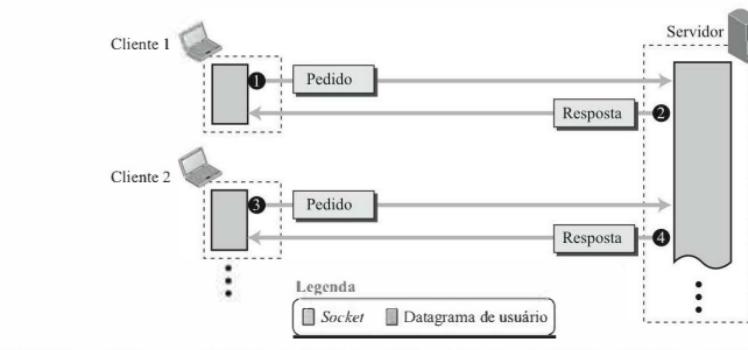



Figura 2.59 *Sockets* para comunicação via UDP.

Diagrama de fluxo da comunicação

A Figura 2.60 mostra um diagrama de fluxo simplificado para a comunicação iterativa. São mostrados vários clientes, mas apenas um servidor. Cada cliente é servido em cada iteração do laço no servidor. Perceba que não há estabelecimento de conexões ou término de conexões. Cada cliente envia e recebe um único datagrama de usuário. Ou seja, se um cliente deseja enviar dois datagramas de usuário, ele é percebido como dois clientes pelo servidor. O segundo datagrama precisa esperar sua vez.

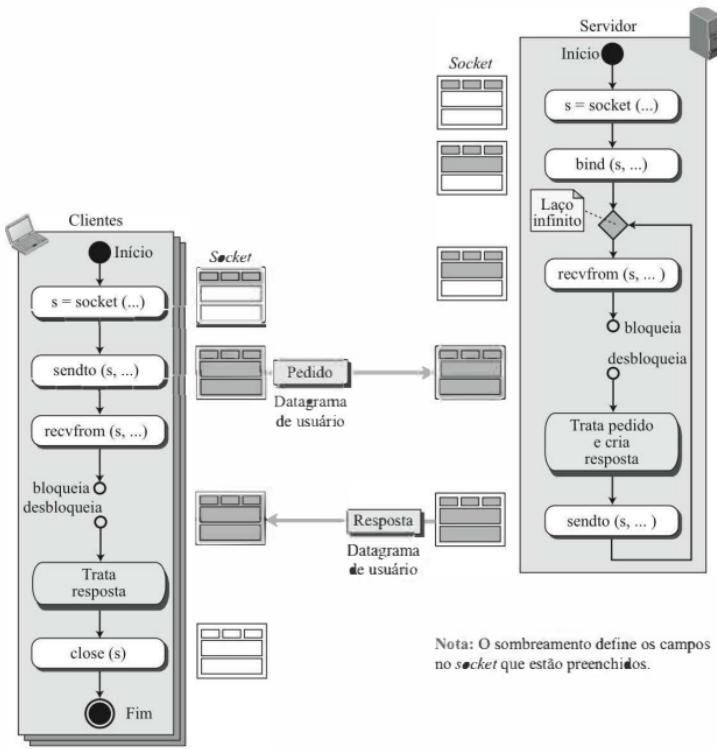



Figura 2.60 Diagrama de fluxo para a comunicação iterativa via UDP.

Processo-servidor O servidor efetua uma *abertura passiva* (*passive open*), de modo que fica pronto para a comunicação, mas espera até que um processo-cliente faça o acesso. Ele invoca a função *socket* para criar um *socket*. Os argumentos passados a essa função preenchem os três primeiros campos do *socket*, mas os campos de endereço local e remoto permanecem indefinidos. Em seguida, o processo-servidor invoca a função *bind* para preencher o campo de endereço de *socket* local (a informação vem do sistema operacional). Ele, então, invoca uma outra função, denominada *recvfrom*, que bloqueia o processo-servidor até que um datagrama de usuário seja recebido. Quando isso acontece, o processo-servidor sai do estado bloqueado e extrai a pedido do datagrama de usuário. Também extrai o endereço de *socket* do remetente para ser usado no passo seguinte. Após o pedido ser processado e a resposta estar pronta, o processo-servidor completa a estrutura de *socket*, preenchendo o campo de endereço remoto com o endereço de *socket* do remetente presente no datagrama de usuário recebido. Agora, o datagrama de usuário está pronto para ser enviado. Isto é feito por meio da invocação a outra função, chamada *sendto*. Perceba que todos os campos do *socket* devem ser preenchidos para que o processo-servidor possa enviar a resposta. Após o envio

da resposta, o processo-servidor inicia uma nova iteração e aguarda pelo acesso de outro cliente. O campo de endereço de *socket* remoto será preenchido novamente com o endereço de um novo cliente (ou do mesmo, mas ele será tratado como um novo cliente). O processo-servidor não tem fim; é executado o tempo todo. O *socket* do servidor nunca é fechado, a não ser que haja algum problema e o processo precise ser abortado.

Processo-cliente O processo-cliente faz uma *abertura ativa* (*active open*), ou seja, ele inicia um acesso, invoca a função *socket* para criar um *socket* e preencher seus três primeiros campos. Embora algumas implementações exijam que o processo-cliente também invoque a função *bind* para preencher o endereço de *socket* local, normalmente isso é feito automaticamente pelo sistema operacional, que seleciona um número temporário de porta para o cliente. O processo-cliente invoca, então, a função *sendto* e fornece as informações referentes ao endereço de *socket* remoto, que deve ser fornecido pelo usuário do processo-cliente. O *socket* está completamente definido nesse momento, e o datagrama de usuário é enviado. O processo-cliente invoca, então, a função *recvfrom*, que bloqueia o processo-cliente até que a resposta volte do processo-servidor. Não há necessidade de obter o endereço de *socket* remoto dessa função porque não há qualquer chamada à função *sendto*. Em outras palavras, a função *recvfrom* no lado do servidor e no lado do cliente se comportam de formas distintas. No processo-servidor, primeiro invocamos a função *recvfrom* e depois a função *sendto*, de modo que o endereço de *socket* remoto usado no *sendto* pode ser obtido a partir de *recvfrom*. Já no processo-cliente, o *sendto* é invocado antes do *recvfrom*, de modo que o endereço remoto deve ser fornecido pelo usuário do programa, que é quem sabe para qual servidor o pedido deve ser enviado. Finalmente, a função *close* é invocada para destruir o *socket*. Perceba que o processo-cliente é finito, ou seja, após a resposta ter sido recebida, o processo-cliente cessa. Embora possamos projetar clientes que enviem múltiplos datagramas de usuário usando um laço, cada iteração do laço será vista com um novo cliente para o servidor.

Exemplos de programação

Nesta seção, vamos mostrar como escrever programas cliente e servidor para simular a aplicação padrão *echo* (eco) usando UDP. O programa-cliente envia uma curta sequência de caracteres ao servidor; o servidor envia uma resposta contendo a mesma sequência de caracteres ao cliente. Essa aplicação costuma ser usada por um computador, o cliente, para testar se outro computador, o servidor, está ativo. Nossas implementações são bastante simples quando comparadas a programas existentes, pois eliminamos alguns detalhes de depuração e verificação de erros por razões de simplicidade.

Programa-servidor de *echo* A Tabela 2.22 mostra o programa-servidor que implementa a funcionalidade de *echo* utilizando UDP. O programa segue o diagrama de fluxo da Figura 2.60.

Tabela 2.22 Programa-servidor de *echo* usando UDP.

1	// Programa-servidor: *echo* usando UDP
2	#include "headerFiles.h"
3	int main (void)
4	{
5	// Declaração e definição de variáveis
6	int s; // Descritor do *socket* (referência)
7	int len; // Tamanho da sequência de caracteres a ser ecoada

(*Continua*)

Tabela 2.22 Programa-servidor de *echo* usando UDP. (*Continuação*)

```
8     char buffer [256];                                    // Buffer de dados
9     struct sockaddr_in servAddr;                          // Endereço de socket do servidor (local)
10    struct sockaddr_in clntAddr;                          // Endereço de socket do cliente (remoto)
11    int clntAddrLen;                                      // Tamanho do endereço de socket do servidor
12    // Constrói endereço de socket local (do servidor)
13    memset (&servAddr, 0, sizeof (servAddr));                       // Aloca memória
14    servAddr.sin_family = AF_INET;                                  // Campo familia
15    servAddr.sin_port = htons (SERVER_PORT)                         // Número de porta padrão
16    servAddr.sin_addr.s_addr = htonl (INADDR_ANY);                  // Endereço IP padrão
17    // Cria socket
18    if ((s = socket (PF_INET, SOCK_DGRAM, 0) < 0);
19    {
20            perror ("Error: socket failed!");
21            exit (1);
22    }
23    // Associa o socket a um endereço e porta locais
24    if ((bind (s, (struct sockaddr*) &servAddr, sizeof (servAddr)) < 0);
25  {
26            perror ("Error: bind failed!");
27            exit (1);
28    }
29    for ( ; ; )// Executa eternamente
30    {
31            // Recebe sequência de caracteres
32            len = recvfrom (s, buffer, sizeof (buffer), 0,
33                        (struct sockaddr*)&clntAddr, &clntAddrLen);
34            // Envia sequência de caracteres
35            sendto (s, buffer, len, 0, (struct sockaddr*)&clntAddr, sizeof(clntAddr));
36    } // Fim do laço "for"
37  } // Fim do programa-servidor de "echo"
```

As linhas 6 a 11 declaram e definem as variáveis usadas no programa. As linhas 13 a 16 alocam memória para o endereço de *socket* do servidor (usando a função *memset*) e preenchem o campo de endereço do *socket* com o valor-padrão fornecido pela camada de transporte. Para inserir o número da porta, usamos a função *htons* (abreviação de *host to network short*), que transforma um valor no formato utilizado pelo *host* para um *short* no formato utilizado pela rede, já que a ordenação de *bytes* nos dois casos pode ser diferente. Para inserir o endereço IP, usamos a função *htonl* (abreviação de *host to network long*), que faz algo semelhante, mas resulta em um *long* e não em um *short*.

As linhas 18 a 22 invocam a função *socket* dentro de uma diretiva *if* para verificar se há erros. Como essa função retorna –1 se a invocação falhar, o programa imprime uma mensagem de erro adequada e finaliza. A função *perror* é uma função de erro padrão em C. Da mesma forma, as linhas 24 a 28 invocam a função *bind* para associar o *socket* ao endereço de *socket* do servidor. Mais uma vez, a função é chamada dentro de uma diretiva *if* para permitir a verificação de erros.

As linhas 29 a 36 usam um laço infinito para permitir que os clientes sejam servidos em cada iteração. As linhas 32 e 33 invocam a função *recvfrom* para ler o pedido enviado pelo cliente. Perceba que essa é uma função bloqueante, ou seja, ela bloqueia o processo até que algum evento ocorra (nesse caso, a recepção de um datagrama de usuário); quando o *socket* recebe o pedido do usuário, a função é desbloqueada, salvando os dados recebidos em *buffer* e, ao mesmo tempo, fornecendo o endereço de *socket* do cliente para completar o último campo ainda não preenchido do *socket*. A linha 35 invoca a função *sendto* para enviar de volta (ecoar) a mesma mensagem para o cliente, usando o endereço de *socket* do cliente obtido da função *recvfrom*. Observe que não é feito qualquer processamento sobre a mensagem de pedido; o servidor apenas ecoa aquilo que ele recebeu.

Programa-cliente de *echo* A Tabela 2.23 mostra o programa-cliente que implementa a funcionalidade *echo* utilizando UDP. O programa segue o diagrama de fluxo da Figura 2.60.

Tabela 2.23 Programa-cliente de *echo* usando UDP.

1	// Programa-cliente: *echo* usando UDP	
2	**#include** "headerFiles.h"	
3	int main (int argc, char* argv[ ])	// Três argumentos a serem verificados mais adiante
4	{	
5	// Declaração e definição de variáveis	
6	int s;	// Descritor do *socket* (referência)
7	int len;	// Tamanho da sequência de caracteres a ser ecoada
8	char* servName;	// Nome do servidor
9	int servPort;	// Porta do servidor
10	char* string;	// Sequência de caracteres a ser ecoada
11	char buffer [256 + 1];	// *Buffer* de dados
12	struct sockaddr_in servAddr;	// Endereço de *socket* do servidor
13	// Verifica e atribui os argumentos passados ao programa	
14	**if** (argc != 3)	
15	{	

(*Continua*)

Tabela 2.23 Programa-cliente de *echo* usando UDP. (*Continuação*)

```
16            printf ("Error: three arguments are needed!");
17            exit(1);
18        }
19        servName = argv[1];
20        servPort = atoi (argv[2]);
21        string = argv[3];
22        // Constrói endereço de socket do servidor
23        memset (&servAddr, 0, sizeof (servAddr));
24        servAddr.sin_family = AF_INET;
25        inet_pton (AF_INET, servName, &servAddr.sin_addr);
26        servAddr.sin_port = htons (servPort);
27        // Cria socket
28        if ((s = socket (PF_INET, SOCK_DGRAM, 0) < 0);
29        {
30            perror ("Error: Socket failed!");
31            exit (1);
32        }
33        // Envia sequência de caracteres a ser ecoada
34        len = sendto (s, string, strlen (string), 0, (struct sockaddr)&servAddr, sizeof (servAddr));
35        // Recebe sequência de caracteres ecoada
36        recvfrom (s, buffer, len, 0, NULL, NULL);
37        // Imprime Print e verifica sequência de caracteres ecoada
38        buffer [len] = '\0';
39        printf ("Echo string received: ";
40        fputs (buffer, stdout);
41        // Fecha o socket
42        close (s);
43        // Finaliza o programa
44        exit (0);
45    } // Fim do programa-cliente de "echo"
```

As linhas 6 a 12 declaram e definem variáveis utilizadas no programa. As linhas 14 a 21 testam e estabelecem os argumentos que são fornecidos quando o programa é executado. Os dois primeiros argumentos fornecem o nome e o número de porta do servidor; o terceiro argumento consiste na sequência de caracteres a ser ecoada. As linhas 23 a 26 alocam memória, convertem o nome do servidor para o endereço IP correspondente usando a função *inet_pton*, que invoca o DNS (discutido anteriormente neste capítulo), e convertem o número da porta para a ordenação de *bytes* adequada. Essas três informações, que são necessárias para a função *sendto*, são armazenadas em variáveis apropriadas.

A linha 34 invoca a função *sendto* para enviar o pedido. A linha 36 chama a função *recvfrom* para receber a mensagem ecoada. Observe que os dois argumentos da mensagem são NULL porque não precisamos extrair o endereço de *socket* do local remoto, já que a mensagem já foi enviada.

As linhas 38 a 40 são utilizadas para exibir a mensagem repetida na tela para fins de depuração. Perceba que, na linha 38, adicionamos um caractere de terminação de *string* no final da mensagem ecoada para torná-la exibível na linha seguinte. Finalmente, a linha 42 fecha o *socket* e a linha 44 sai do programa.

Comunicação usando TCP

Conforme descrito anteriormente, o TCP é um protocolo orientado à conexão. Antes de enviar ou receber dados, uma conexão precisa ser estabelecida entre o cliente e o servidor. Depois que a conexão é estabelecida, as duas partes podem trocar dados entre si, desde que os tenham. A comunicação TCP pode ser iterativa (servindo um cliente de cada vez) ou concorrente (servindo vários clientes ao mesmo tempo). Nesta seção, discutimos apenas a abordagem iterativa. Mostraremos a abordagem concorrente em Java no Capítulo 11.

Sockets usados no TCP

O servidor TCP usa dois *sockets* distintos, um para o estabelecimento da conexão e outro para a transferência de dados. Chamamos o primeiro de *socket de escuta* (ou *listen socket*) e o segundo simplesmente de *socket*. O objetivo de se ter dois tipos de *socket* é separar a fase de conexão da fase de troca de dados. Um servidor usa um *socket* de escuta para ficar escutando novos clientes tentando estabelecer conexões. Quando uma conexão é estabelecida, o servidor cria um *socket* para trocar dados com o cliente e, finalmente, encerra a conexão. O cliente usa apenas um *socket* para o estabelecimento da conexão e troca de dados (ver Figura 2.61).

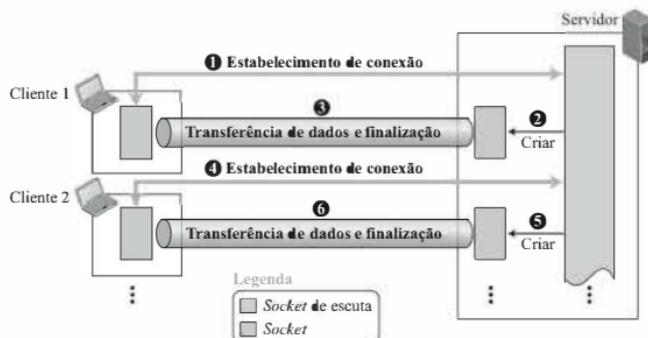



Figura 2.61 *Sockets* usados em uma comunicação TCP.

Diagrama de fluxo da comunicação

A Figura 2.62 mostra um diagrama de fluxo simplificado da comunicação iterativa. A figura mostra vários clientes, mas apenas um servidor. Cada cliente é servido em uma das iterações do laço. O diagrama de fluxo é quase igual ao usado pelo UDP, mas existem diferenças que explicamos para cada lado da comunicação.

Processo-servidor Na Figura 2.62, o processo-servidor TCP, tal como o processo-servidor UDP, invoca as funções *socket* e *bind*, mas as duas funções criam o *socket* de escuta para ser usado apenas para a fase de estabelecimento da conexão. O processo do servidor, em seguida, invoca a função *listen* para permitir que o sistema operacional *comece* a aceitar os clientes, completando a fase de conexão e colocando-os em uma lista de espera para serem servidos. Essa função também define o tamanho da lista de espera de clientes conectados, a qual depende da complexidade do processo--servidor, mas normalmente assume o valor 5.

O processo do servidor, então, inicia um processo iterativo para servir os clientes um a um. Em cada iteração, o processo-servidor invoca a função *accept*, que remove um cliente da lista de espera de clientes conectados para servi-lo. Se a lista estiver vazia, a função *accept* fica bloqueada até que haja um cliente para ser servido. Quando a função *accept* retorna, ela cria um novo *socket* igual ao *socket* de escuta, que passa, então, a executar em segundo plano (*background*) e o novo *socket* entra em atividade. O processo-servidor usa o endereço do *socket* cliente obtido durante o estabelecimento da conexão para preencher o campo de endereço de *socket* remoto do *socket* recém-criado.

A partir de então, o cliente e o servidor podem trocar dados. Não mostramos a forma específica em que a transferência de dados ocorre porque isso depende do par cliente/servidor em questão. O TCP usa as funções *send* e *recv* para transferir *bytes* de dados entre eles. Essas duas funções são mais simples do que as funções *sendto* e *recvfrom* utilizadas no UDP, porque não fornecem o endereço do *socket* remoto, já que uma conexão foi estabelecida entre o cliente e o servidor. No entanto, uma vez que o TCP é utilizado para transferir um número ilimitado de mensagens, cada aplicação precisa projetar cuidadosamente o processo de transferência de dados. As funções *send* e *recv* podem ser invocadas várias vezes para que seja possível transferir uma grande quantidade de dados. O diagrama de fluxo da Figura 2.62 pode ser considerado genérico; para cada finalidade específica, o diagrama para a caixa *transferência de dados do servidor* deve ser definido. Faremos isso para um exemplo simples, quando discutirmos o programa cliente/servidor da função *echo*.

Processo-cliente O diagrama de fluxo dos clientes é bastante semelhante à sua versão em UDP, exceto pelo fato de que a caixa *transferência de dados do cliente* precisa ser definida para cada caso específico. Faremos isso quando formos escrever um programa específico mais adiante.

Exemplos de programação

Nesta seção, vamos mostrar como podemos escrever programas cliente e servidor que simulem a aplicação padrão de *echo* usando TCP. O programa-cliente envia uma curta sequência de caracteres ao servidor; este envia uma resposta contendo a mesma sequência de caracteres ao cliente. Entretanto, antes de fazer isto, precisamos definir os diagramas de fluxo para as caixas de transferência de dados do cliente e do servidor, que são mostrados na Figura 2.63.

Para esse caso especial, já que o tamanho da sequência de caracteres a ser enviada é pequena (apenas algumas palavras), podemos fazê-lo com uma única chamada à função *send* no lado do cliente. No entanto, não existem garantias de que o TCP enviará a mensagem completa em um só segmento. Portanto, precisamos usar um conjunto de chamadas à função *recv* no lado do servidor (em um laço), para receber a mensagem completa e armazená-la em um *buffer* para que ela possa ser enviada de volta de uma só vez. Quando o servidor está enviando a mensagem ecoada, ele também pode usar vários segmentos para fazê-lo, o que significa que a função recv no lado do cliente precisa ser chamada tantas vezes quanto for necessária.

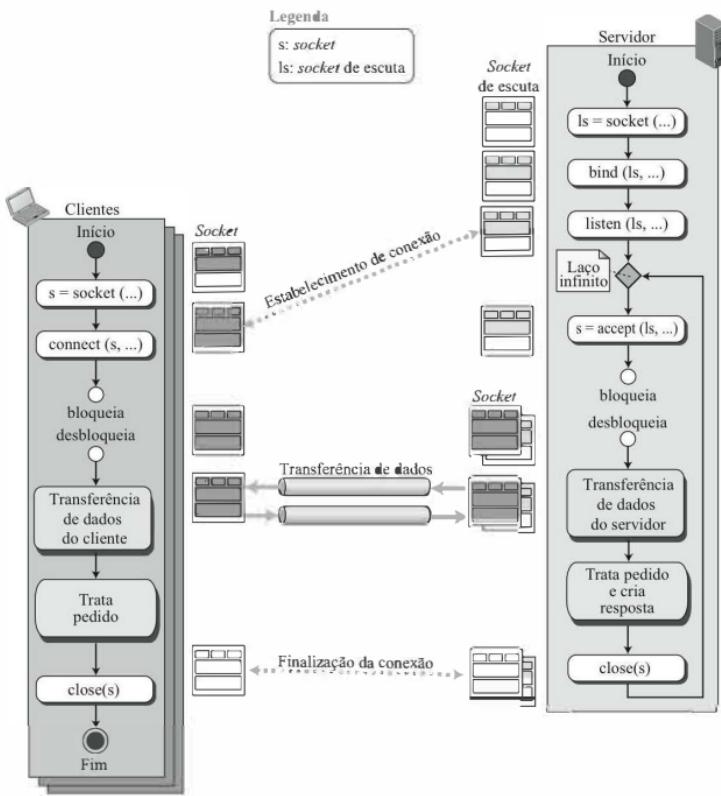



Figura 2.62 Diagrama de fluxo para uma comunicação TCP iterativa.

Outro problema a ser resolvido é a criação dos *buffers* que armazenam dados em cada lado da comunicação. Precisamos controlar quantos *bytes* de dados foram recebidos e onde o próximo conjunto de dados deve ser armazenado. O programa define algumas variáveis para controlar a situação, conforme mostra a Figura 2.64. Em cada iteração, o ponteiro (ptr) avança para apontar para os próximos *bytes* a serem recebidos, o comprimento dos *bytes* recebidos (len) aumenta e o número máximo de *bytes* a serem recebidos (maxlen) diminui.

Após essas duas considerações, podemos, então, escrever os programas servidor e cliente.

Programa-servidor de *echo* A Tabela 2.24 mostra o programa-servidor da função *echo* utilizando TCP. O programa segue o diagrama de fluxo da Figura 2.62.

As linhas 6 a 16 declaram e definem variáveis. As linhas 18 a 21 alocam memória e constroem o endereço do *socket* local (servidor) como descrito no caso do UDP. As linhas 23 a 27

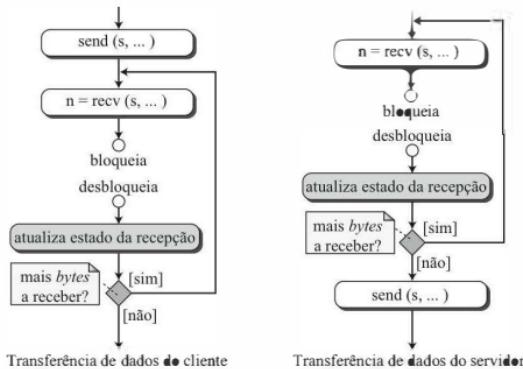



Figura 2.63 Diagrama de fluxo para as caixas de transferência de dados do cliente e do servidor.

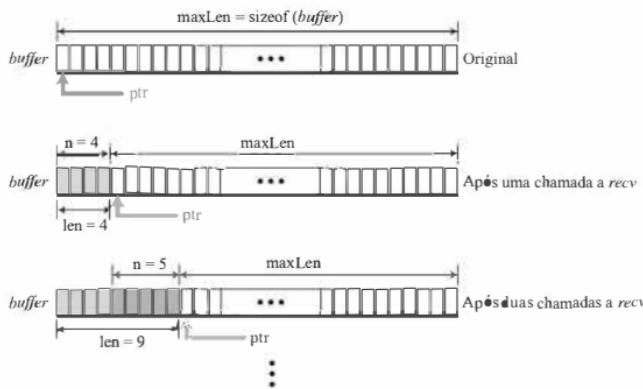



Figura 2.64 *Buffer* usado para recepção.

criam o *socket* de escuta. As linhas 29 a 33 associam o *socket* de escuta ao endereço do *socket* servidor construído nas linhas 18 a 21. As linhas 35 a 39 são novas, introduzidas pela comunicação TCP. A função *listen* é invocada para permitir que o sistema operacional conclua a fase de estabelecimento da conexão e coloque os clientes na lista de espera. As linhas 44 a 48 invocam a função *accept* para remover o próximo cliente da lista de espera e começar a servi-lo. Essa função bloqueia o processo caso não haja clientes na lista de espera. As linhas 50 a 56 correspondem ao código da seção de transferência de dados representado na Figura 2.63. O tamanho do *buffer* máximo, que corresponde ao comprimento da cadeia de caracteres ecoada, é o mesmo mostrado na Figura 2.64.

Tabela 2.24 Programa-servidor de *echo* usando os serviços do TCP.

1	// Programa-servidor: *echo* usando TCP	
2	**#include** "headerFiles.h"	
3	**int** main (**void**)	
4	{	
5	// Declaração e definição de variáveis	
6	**int** ls;	// Descritor do *socket* de escuta (referência)
7	**int** s;	// Descritor do *socket* (referência)
8	**char** buffer [256]; // Buffer de dados	
9	**char*** ptr = buffer;	// Ponteiro para ser movido sobre o *buffer* de dados
10	**int** len = 0;	// Número de *bytes* a serem enviados ou recebidos
11	**int** maxLen = sizeof (buffer);	// Número máximo de *bytes* a serem recebidos
12	**int** n = 0;	// Número de *bytes* em cada chamada a *recv*
13	**int** waitSize = 16;	// Tamanho da lista de espera de clientes
14	**struct** sockaddr_in serverAddr;	// Endereço do servidor
15	**struct** sockaddr_in clientAddr;	// Endereço do cliente
16	**int** clntAddrLen;	// Tamanho do endereço do cliente
17	// Cria endereço de *socket* local (servidor)	
18	memset (&servAddr, 0, sizeof (servAddr));	
19	servAddr.sin_family = AF_INET;	
20	servAddr.sin_addr.s_addr = htonl (INADDR_ANY);	// Endereço IP padrão
21	servAddr.sin_port = htons (SERV_PORT);	// Porta-padrão
22	// Cria *socket* de escuta	
23	**if** (ls = socket (PF_INET, SOCK_STREAM, 0) < 0);	
24	{	
25	perror ("Error: Listen socket failed!");	
26	exit (1);	
27	}	
28	// Associa o *socket* de escuta ao endereço de *socket* local	
29	**if** (bind (ls, &servAddr, sizeof (servAddr)) < 0);	
30	{	

(*Continua*)

Tabela 2.24 Programa-servidor de *echo* usando os serviços do TCP. (*Continuação*)

31	perror ("Error: binding failed!");
32	exit (1);
33	}
34	// Escuta por pedidos de conexão
35	**if** (listen (ls, waitSize) < 0);
36	{
37	perror ("Error: listening failed!");
38	exit (1);
39	}
40	// Trata a conexão
41	**for** (; ;) // Executa infinitamente
42	{
43	// Aceita conexões do cliente
44	**if** (s = accept (ls, &clntAddr, &clntAddrLen) < 0);
45	{
46	perror ("Error: accepting failed!);
47	exit (1);
48	}
49	// Seção de transferência de dados
50	**while** ((n = recv (s, ptr, maxLen, 0)) > 0)
51	{
52	ptr + = n; // Move ponteiro ao longo do *buffer*
53	maxLen - = n; // Ajusta número máximo de *bytes* a receber
54	len + = n; // Atualiza número de *bytes* recebidos
55	}
56	send (s, buffer, len, 0); // Envia de volta (ecoa) todos os *bytes* recebidos
57	// Fecha o *socket*
58	close (s);
59	} // Fim do laço for
60	} // Fim do programa-servidor de *echo*

Programa-cliente de *echo* A Tabela 2.25 mostra o programa-cliente da função *echo* utilizando TCP. O programa segue o diagrama de fluxo básico da Figura 2.62.

Tabela 2.25 Programa-cliente de *echo* usando TCP

1	// Programa-cliente: *echo* usando TCP	
2	**#include** "headerFiles.h"	
3	**int** main (int argc, char* argv[ ])	// Três argumentos a serem verificados mais adiante
4	{	
5	// Declaração e definição de variáveis	
6	**int** s;	// Descritor do *socket*
7	**int** n;	// Número de *bytes* em cada chamada a *recv*
8	**char*** servName;	// Nome do servidor
9	**int** servPort;	// Número de porta do servidor
10	**char*** string;	// Sequência de caracteres a ser ecoada
11	**int** len;	// Tamanho da sequência de caracteres a ser ecoada
12	**char** buffer [256 + 1];	// *Buffer*
13	**char*** ptr = buffer;	// Ponteiro para ser movido sobre o *buffer* de dados
14	**struct** sockaddr_in serverAddr;	// Endereço de *socket* do servidor
15	// Verifica e assigna os argumentos passados	
16	**if** (argc != 3)	
17	{	
18	printf ("Error: three arguments are needed!");	
19	exit (1);	
20	}	
21	servName = arg [1];	
22	servPort = atoi (arg [2]);	
23	string = arg [3];	
24	// Cria endereço de *socket* remoto (servidor)	
25	memset (&servAddr, 0, sizeof(servAddr);	
26	serverAddr.sin_family = AF_INET;	
27	inet_pton (AF_INET, servName, &serverAddr.sin_addr);	// Endereço IP do servidor

(*Continua*)

Tabela 2.25 Programa-cliente de *echo* usando TCP. (*Continuação*)

28	serverAddr.sin_port = htons (servPort);	// Endereço de porta do servidor
29	// Cria *socket*	
30	**if** ((s = socket (PF_INET, SOCK_STREAM, 0) < 0);	
31	{	
32	perror ("Error: socket creation failed!");	
33	exit (1);	
34	}	
35	// Conecta ao servidor	
36	**if** (connect (sd, (struct sockaddr*)&servAddr, sizeof(servAddr)) < 0);	
37	{	
38	perror ("Error: connection failed!");	
39	exit (1);	
40	}	
41	// Seção de transferência de dados	
42	send (s, string, strlen(string), 0);	
43	**while** ((n = recv (s, ptr, maxLen, 0)) > 0)	
44	{	
45	ptr + = n;	// Move ponteiro ao longo do *buffer*
46	maxLen - = n;	// Ajusta número máximo de *bytes*
47	len += n;	// Atualiza número de *bytes* recebidos
48	} // Fim do laço while	
49	// Imprime e verifica a sequência de caracteres ecoada	
50	buffer [len] = '\0';	
51	printf ("Echoed string received: ");	
52	fputs (buffer, stdout);	
53	// Fecha o *socket*	
54	close (s);	
55	// Encerra o programa	
56	exit (0);	
57	} // Fim do programa-cliente de *echo*	

O programa-cliente usando TCP é muito parecido com o programa-cliente que usa UDP, com algumas poucas diferenças. Como o TCP é um protocolo orientado à conexão, a função *connect* é invocada nas linhas 36 a 40 para fazer a conexão com o servidor. A transferência de dados é realizada nas linhas 42 a 48, utilizando a ideia representada na Figura 2.63. O comprimento dos dados recebidos e o movimento do ponteiro são tratados conforme mostrado na Figura 2.64.

2.6 MATERIAL DO FINAL DO CAPÍTULO

2.6.1 Leitura adicional

Para obter mais detalhes sobre os assuntos discutidos neste capítulo, recomendamos os seguintes livros e RFCs. Os itens entre colchetes referem-se à lista de referências no final do livro.

Livros

Diversos livros dão uma cobertura bastante completa dos materiais discutidos neste capítulo, incluindo [Com 06], [Mir 07], [Ste 94], [Tan 03], [Bar *et al.* 05].

RFCs

O HTTP é discutido nas RFCs 2068 e 2109. O FTP é discutido nas RFCs 959, 2577 e 2585. O TELNET é discutido nas RFCs 854, 855, 856, 1041, 1091, 1372 e 1572. O SSH é discutido nas RFCs 4250, 4251, 4252, 4253, 4254 e 4344. O DNS é discutido nas RFCs 1034, 1035, 1996, 2535, 3008, 3658, 3755, 3757, 3845, 3396 e 3342. O SMTP é discutido nas RFCs 2821e 2822. O POP3 é explicado na RFC 1939. O MIME é discutido nas RFCs 2046, 2047, 2048 e 2049.

2.6.2 Termos-chave

- acesso local
- acesso remoto
- Agente de Acesso a Mensagens (MAA – Message Access Agent)
- Agente de Transferência de Mensagens (MTA – Mail Transfer Agent)
- Agente de Usuário (UA – User Agent)
- Chord
- conexão não persistente
- conexão persistente
- *cookie*
- documento ativo (*active document*)
- documento dinâmico
- documento estático
- domínio de país
- domínio genérico
- encaminhamento de porta (*port forwarding*)
- endereço de *socket*
- espaço de nomes (*name space*)
- espaço de nomes de domínio (*domain name space*)
- Extensões Multifunção para Mensagens de Internet (MIME – Multipurpose Internet Mail Extensions)
- hipermídia
- hipertexto
- identificador (*label*)
- Interface de Camada de Transporte (TLI – Transport Layer Interface)
- Interface de Programação de Aplicativos (API – Application Programming Interface)
- interface *socket*
- Kademlia
- Localizador Uniforme de Recursos (URL – Uniform Resource Locator)
- navegador (*browser*)
- nome de domínio
- Nome de Domínio Parcialmente Qualificado (PQDN – Partially Qualified Domain Name)

- Nome de Domínio Totalmente Qualificado (FQDN – Fully Qualified Domain Name)
- página Web
- paradigma cliente-servidor
- paradigma *peer-to-peer* (par a par ou P2P)
- Pastry
- Protocolo de Acesso a Correio da Internet, versão 4 (IMAP4 – Internet Mail Access Protocol, version 4)
- Protocolo de Agência de Correio versão 3 (POP3 – Post Office Protocol version 3)
- Protocolo de Transferência de Arquivos (FTP – File Transfer Protocol)
- Protocolo de Transferência de Hipertexto (HTTP – HyperText Transfer Protocol)
- Protocolo Simples de Transferência de Correio (SMTP – Simple Mail Transfer Protocol)
- resolução iterativa
- resolvedor
- servidor *proxy*
- servidor raiz
- Secure Shell (SSH)
- Sistema de Nomes de Domínio (DNS – Domain Name System)
- Sistema de Nomes de Domínio Dinâmico (DDNS – Dynamic Domain Name System)
- STREAM
- Tabela de *Hash* Distribuída (DHT – Distributed Hash Table)
- Terminal de Rede (TELNET – TerminaL Network)
- Terminal Virtual de Rede (NVT – Network Virtual Terminal)
- World Wide Web (WWW)
- zona

2.6.3 Resumo

As aplicações na Internet são projetadas usando o paradigma cliente-servidor ou o paradigma *peer-to-peer*. No paradigma cliente-servidor, um aplicativo, denominado servidor, fornece serviços, enquanto outro, denominado cliente, recebe os serviços. Um programa-servidor executa indefinidamente; um programa-cliente tem um tempo de vida finito. No paradigma *peer-to-peer*, um *peer* (um nó da rede) pode atuar tanto como um cliente quanto como um servidor.

A World Wide Web (WWW) é um repositório de informações ligadas entre si a partir de pontos espalhados pelo mundo inteiro. Documentos de hipertexto e de hipermídia são ligados uns aos outros por meio de ponteiros. O Protocolo de Transferência de Hipertexto (HTTP – HyperText Transfer Protocol) é o principal protocolo usado para acessar os dados disponíveis na World Wide Web (WWW).

O Protocolo de Transferência de Arquivos (FTP – File Transfer Protocol) é uma aplicação cliente-servidor que usa a pilha de protocolos TCP/IP para copiar arquivos de um *host* para outro. O FTP requer duas conexões para a transferência de dados: uma de controle e uma de dados. O FTP utiliza o conjunto de caracteres NVT ASCII para permitir a comunicação entre sistemas diferentes.

O correio eletrônico (e-mail) é uma das aplicações mais comuns na Internet. A arquitetura do e-mail é composta por diversos componentes, como o Agente de Usuário (UA – User Agent), o Agente de Transferência de Correio (MTA – Mail Transfer Agent), e o Agente de Acesso a Mensagens (MAA – Message Access Agent). O protocolo que implementa o MTA é chamado Protocolo Simples de Transferência de Correio (SMTP – Simple Mail Transfer Protocol). Dois protocolos são utilizados para implementar o MAA: o Protocolo de Agência de Correio versão 3 (POP3 – Post Office Protocol version 3) e o Protocolo de Acesso a Correio da Internet versão 4 (IMAP4 – Internet Mail Access Protocol version 4).

O TELNET é uma aplicação cliente-servidor que permite que um usuário efetue o *login* em um computador remoto, dando ao usuário acesso ao sistema remoto. No acesso via TELNET, isso é semelhante a um ambiente de tempo compartilhado.

O Sistema de Nomes de Domínio (DNS – Domain Name System) é uma aplicação cliente-servidor que identifica cada *host* na Internet por um nome unívoco. O DNS organiza o espaço de nomes em uma estrutura hierárquica com o objetivo de descentralizar as responsabilidades envolvidas no processo de definir os nomes dos *hosts*.

Em uma rede *peer-to-peer* (par a par, ou P2P), os usuários da Internet que desejam compartilhar seus recursos tornam-se *peers* e formam uma rede. Redes *peer-to-peer* podem ser classificadas como centralizadas e descentralizadas. Em uma rede P2P centralizada, o sistema de diretórios usa o paradigma cliente-servidor, mas o armazenamento e o *download* dos arquivos são feitos usando o paradigma *peer-to-peer*. Em uma rede descentralizada, tanto o sistema de diretórios como os processos de armazenamento e transferência de arquivos usam o paradigma *peer-to-peer*.

2.7 ATIVIDADES PRÁTICAS

2.7.1 Testes

Diversos testes interativos relativos a este capítulo podem ser encontrados no *site* www.mhhe.com/forouzan. Recomendamos que o estudante realize os testes para verificar o seu entendimento do material apresentado antes de continuar com as atividades práticas.

Questões

2-1 Suponha que adicionemos um novo protocolo à camada de aplicação. Quais mudanças precisamos fazer nas outras camadas?

2-2 Explique qual entidade fornece serviços e qual recebe serviços no paradigma cliente-servidor.

2-3 No paradigma cliente-servidor, explique por que um servidor deve ser executado o tempo todo, mas um cliente pode ser executado apenas quando necessário.

2-4 Um programa escrito para utilizar os serviços do UDP pode ser executado em um computador que tenha instalado apenas o TCP como protocolo da camada de transporte? Explique.

2-5 No fim de semana, Alice normalmente precisa acessar arquivos armazenados no computador do escritório a partir do *laptop* em sua casa. Na última semana, ela instalou uma cópia de um servidor FTP no computador do escritório e uma cópia de um cliente FTP em seu *laptop*, em casa. Ela ficou decepcionada quando não conseguiu acessar seus arquivos durante o fim de semana. O que pode ter dado errado?

2-6 A maioria dos sistemas operacionais instalados em computadores pessoais vem com vários aplicativos que funcionam como clientes, mas normalmente não vem com aplicativos do tipo servidor. Explique a razão.

2-7 Uma nova aplicação precisa ser projetada usando o paradigma cliente-servidor. Se o cliente e o servidor precisam trocar apenas mensagens curtas entre si, sem se preocupar com perdas ou corrupção de mensagens, qual protocolo da camada de transporte você recomenda?

2-8 Qual (ou quais) dos seguintes elementos pode(m) funcionar como entidade(s) de entrada de dados?

a. Um teclado

b. Um monitor

c. Um *socket*

2-9 Um endereço de *socket* de origem é uma combinação de um endereço IP com um número de porta. Explique o que cada um desses elementos identifica.

2-10 Explique como um processo-cliente encontra o endereço IP e o número de porta a ser utilizado como endereço de *socket* remoto.

2-11 Se a resposta a uma solicitação HTTP exige a execução de um programa no lado do servidor de modo que o resultado é enviado ao cliente, esse programa é um exemplo de

a. Um documento estático.

b. Um documento dinâmico.

c. Um documento ativo.

2-12 Considere que criamos um novo aplicativo cliente-servidor que exige o uso de conexões persistentes. Podemos usar o UDP como

protocolo de camada de transporte subjacente para esse novo aplicativo?

2-13 Alice tem um clipe de vídeo que Bob gostaria de receber e Bob tem um outro clipe de vídeo que Alice gostaria de receber. Bob cria uma página Web e executa um servidor HTTP. Como Alice pode obter o clipe de Bob? Como Bob pode obter o clipe de Alice?

2-14 Quando um servidor HTTP recebe uma mensagem de solicitação de um cliente HTTP, como o servidor sabe o instante em que todos os cabeçalhos chegaram e que o corpo da mensagem virá a seguir?

2-15 Em uma conexão HTTP não persistente, como o HTTP pode informar ao protocolo TCP que o final da mensagem foi atingido?

2-16 Você consegue identificar uma analogia em nosso cotidiano de uma situação em que usamos duas conexões separadas para nos comunicarmos, de modo similar às conexões de controle e de dados no FTP?

2-17 O FTP usa dois números de porta bem conhecidos, distintos, para as conexões de controle e dados. Isto significa que duas conexão TCP separadas são criadas para a troca de informações de controle e de dados?

2-18 O FTP usa os serviços do TCP para a troca de informações de controle e para a transferência de dados. O FTP poderia ter utilizado os serviços de UDP para alguma dessas duas conexões? Explique.

2-19 No FTP, qual entidade (cliente ou servidor) inicia (abre ativamente) a conexão de controle? Qual entidade inicia (abre ativamente) a conexão de transferência de dados?

2-20 O que você acha que aconteceria se a conexão de controle fosse cortada antes do final de uma sessão FTP? Isto afetaria a conexão de dados?

2-21 No FTP, se o cliente precisar recuperar um arquivo proveniente do servidor e armazenar um arquivo no lado do servidor, quantas conexões de controle e quantas conexões de transferência de dados são necessárias?

2-22 No FTP, um servidor pode recuperar um arquivo proveniente do cliente?

2-23 No FTP, um servidor pode obter a lista dos arquivos ou dos diretórios do cliente?

2-24 O FTP pode transferir arquivos entre dois *hosts* que usam sistemas operacionais distintos, com diferentes formatos de arquivo. Qual é a razão?

2-25 O FTP adota um formato de mensagens para a troca de comandos e respostas durante a conexão de controle?

2-26 O FTP adota um formato de mensagem para trocar arquivos ou uma lista de diretórios/arquivos durante a conexão de transferência de arquivos?

2-27 Podemos ter uma conexão de controle sem uma conexão de transferência de dados no FTP? Explique.

2-28 Podemos ter uma conexão de transferência de dados sem uma conexão de controle no FTP? Explique.

2-29 Considere que precisamos obter um conteúdo de áudio usando o FTP. Que tipo de arquivo devemos especificar no nosso comando?

2-30 O HTTP e o FTP podem recuperar um arquivo de um servidor. Qual protocolo devemos usar para obter um arquivo?

2-31 Os comandos HELO e MAIL FROM são necessários no SMTP? Por quê?

2-32 Na Figura 2.20 apresentada no texto, qual é a diferença entre o MAIL FROM no envelope e o FROM no cabeçalho?

2-33 Alice fez uma longa viagem, durante a qual não verificou seus e-mails. Ela descobre que perdeu alguns e-mails ou anexos que seus amigos afirmam ter enviado. Qual pode ser o problema?

2-34 Considere que um cliente TELNET use ASCII para representar caracteres, mas o servidor TELNET use EBCDIC para a mesma finalidade. Como o cliente pode acessar o servidor quando as representações de caracteres forem diferentes?

2-35 O aplicativo TELNET não tem comandos como aqueles encontrados no FTP ou no HTTP para permitir que o usuário faça algo, como transferir um arquivo ou acessar uma página Web. Como esse aplicativo pode ser útil?

2-36 Um *host* pode usar um cliente TELNET para utilizar os serviços fornecidos por outras aplicações cliente-servidor, como FTP ou HTTP?

2-37 No DNS, quais dos itens a seguir são FQDNs e quais são PQDNs?

a. xxx

b. xxx.yyy.net

c. zzz.yyy.xxx.edu.

2-38 Em uma rede baseada em DHT, considere que $m = 4$. Se o *hash* de um identificador de

nó é 18, qual é a localização do nó no espaço de endereços da DHT?

2-39 Em uma rede baseada em DHT, suponha que o nó 4 tem um arquivo com a chave 18. O nó sucessor mais próximo da chave 18 é o nó 20. Onde o arquivo será armazenado no:

a. método direto?

b. método indireto?

2-40 Em uma rede Chord, temos um nó N5 e uma chave c5. O nó N5 é o predecessor de c5? O nó N5 é o sucessor de c5?

2-41 Em uma rede Kademlia, o tamanho do espaço de identificadores é 1024. Qual é a altura da árvore binária (a distância entre a raiz e cada folha)? Qual é o número de folhas? Qual é o número de subárvores para cada nó? Qual é o número de linhas em cada tabela de roteamento?

2-42 No Kademlia, considere que $m = 4$ e que os nós ativos sejam N4, N7 e N12. Onde a chave c3 é armazenada nesse sistema?

Problemas

2-1 Suponha que haja um servidor com o nome de domínio *www.comum.com*.

a. Construa uma solicitação HTTP para recuperar o documento */usr/users/doc*. O cliente aceita MIME versão 1, imagens GIF ou JPEG, mas o documento não pode ser mais antigo do que quatro dias.

b. Mostre a resposta HTTP para o item *a* no caso de um pedido bem-sucedido.

2-2 No HTTP, desenhe uma figura que mostre a aplicação de *cookies* em um cenário no qual o servidor permite acesso apenas a clientes cadastrados.

2-3 No HTTP, desenhe uma figura que mostre a aplicação de *cookies* em um portal Web que usa dois *sites*.

2-4 No HTTP, desenhe uma figura que mostre a aplicação de *cookies* em um cenário em que o servidor utiliza *cookies* para propaganda. Use apenas três *sites*.

2-5 Desenhe um diagrama que mostre o uso de um servidor *proxy* que faça parte da rede do cliente:

a. Mostre as transações entre o cliente, o servidor *proxy* e o servidor-alvo quando a resposta está armazenada no servidor *proxy*.

b. Mostre as transações entre o cliente, o servidor *proxy* e o servidor-alvo quando a resposta não está armazenada no servidor *proxy*.

2-6 No Capítulo 1, mencionamos que a pilha de protocolos TCP/IP, diferentemente do modelo OSI, não tem uma camada de apresentação. Mas um protocolo da camada de aplicação pode incluir algumas das funcionalidades definidas nessa camada, caso necessário. O HTTP tem alguma das funcionalidades da camada de apresentação?

2-7 O HTTP versão 1.1 define o uso de conexões persistentes por padrão. Consultando a RFC 2616, descreva como um cliente ou um servidor pode modificar essa opção-padrão para a alternativa não persistente.

2-8 No Capítulo 1, mencionamos que a pilha de protocolos TCP/IP, diferentemente do modelo OSI, não tem camada de sessão. Mas um protocolo da camada de aplicação pode incluir algumas das funcionalidades definidas nessa camada, caso necessário. O HTTP tem alguma das funcionalidades da camada de sessão?

2-9 No SMTP, o remetente envia texto não formatado. Mostre o cabeçalho MIME correspondente.

2-10 No SMTP,

a. uma mensagem não ASCII de 1.000 *bytes* é codificada usando base64. Quantos *bytes* aparecem na mensagem codificada? Quantos *bytes* são redundantes? Qual é a proporção de *bytes* redundantes em relação à mensagem inteira?

b. uma mensagem de 1.000 *bytes* é codificado usando codificação *quoted printable*. A constituição da mensagem é de 90% de caracteres ASCII e 10% de caracteres não ASCII. Quantos *bytes* aparecem na mensagem codificada? Quantos *bytes* são redundantes? Qual é a proporção de *bytes* redundantes em relação à mensagem inteira?

c. compare os resultados dos dois casos anteriores. Quanto se ganha em eficiência

quando a mensagem é uma combinação de caracteres ASCII e não ASCII?

2-11 Codifique a seguinte mensagem em base64:

01010111 00001111 11110000

2-12 Codifique a seguinte mensagem usando codificação *quoted-printable*:

01001111 10101111 01110001

2-13 De acordo com a RFC 1939, uma sessão POP3 pode estar em um dos seguintes quatro estados: fechada, autorização, transação ou atualização. Desenhe um diagrama mostrando os quatro estados e como o POP3 faz a transição entre eles.

2-14 O protocolo POP3 apresenta alguns comandos básicos (que cada cliente/servidor precisa implementar). Usando as informações presentes na RFC 1939, descreva o significado e o uso dos seguintes comandos básicos:

a. STAT **b.** LIST **c.** DELE 4

2-15 O protocolo POP3 apresenta alguns comandos opcionais (que cada cliente/servidor pode implementar). Usando as informações presentes na RFC 1939, descreva o significado e o uso dos seguintes comandos opcionais:

a. UIDL

b. TOP 1 15

c. USER

d. PASS

2-16 Usando RFC 1939, considere que um cliente POP3 está no modo de obter-e-manter (*download-and-keep*). Mostre a interação entre o cliente e o servidor se o cliente tem apenas duas mensagens de 192 e 300 *bytes* para obter do servidor.

2-17 Usando RFC 1939, considere que um cliente POP3 está no modo de obter-e-remover (*download-and-delete*). Mostre a interação entre o cliente e o servidor se o cliente tem apenas duas mensagens de 230 e 400 *bytes* para obter do servidor.

2-18 No Capítulo 1, mencionamos que a pilha de protocolos TCP/IP, diferentemente do modelo OSI, não tem uma camada de apresentação. Mas um protocolo da camada de aplicação pode incluir algumas das funcionalidades definidas nessa camada, caso necessário. O SMTP apresenta alguma das funcionalidades da camada de apresentação?

2-19 No Capítulo 1, mencionamos que a pilha de protocolos TCP/IP, diferentemente do modelo OSI, não tem uma camada de sessão. Mas um protocolo da camada de aplicação pode incluir algumas das funcionalidades definidas nessa camada, caso necessário. O SMTP e o POP3 apresentam alguma das funcionalidades da camada de sessão?

2-20 No FTP, suponha que um cliente cujo nome de usuário seja John precisa armazenar um videoclipe chamado *video2* no diretório */top/videos/geral* do servidor. Mostre os comandos e as respostas trocadas entre o cliente e o servidor se o cliente escolher o número de porta efêmera 56002.

2-21 No FTP, um usuário (Jânio) quer recuperar um arquivo no formato EBCDIC chamado *enorme*, o qual se encontra no diretório */usr/users/relatorio*, usando a porta efêmera 61017. O arquivo é tão grande que o usuário gostaria de comprimi-lo antes de transferi-lo. Mostre todos os comandos e respostas envolvidos nessa tarefa.

2-22 No FTP, um usuário (Marcos) quer criar uma nova pasta chamada *Marcos* no diretório */usr/usrs/cartas*. Mostre todos os comandos e respostas envolvidos nessa tarefa.

2-23 No FTP, uma usuária (Maria) quer mover um arquivo chamado *arquivo1* do diretório */usr/users/relatorio* para o diretório */usr/top/cartas*. Ela também deseja renomear o arquivo. Primeiro, precisamos fornecer o nome do arquivo antigo e, em seguida, definir o novo nome. Mostre todos os comandos e respostas envolvidos nessa tarefa.

2-24 No Capítulo 1, mencionamos que a pilha de protocolos TCP/IP, diferentemente do modelo OSI, não tem uma camada de apresentação. Mas um protocolo da camada de aplicação pode incluir algumas das funcionalidades definidas nessa camada, caso necessário. O FTP apresenta alguma das funcionalidades da camada de apresentação?

2-25 No Capítulo 1, mencionamos que a pilha de protocolos TCP/IP, diferentemente do modelo OSI, não tem uma camada de sessão. Mas um protocolo da camada de aplicação pode incluir algumas das funcionalidades definidas nessa camada, caso necessário. O FTP apresenta alguma das funcionalidades da camada de sessão?

2-26 No Chord, considere que o tamanho do espaço de identificadores seja 16. Os nós ativos são N3, N6, N8 e N12. Mostre a tabela de

derivação (apenas a chave-alvo e a coluna de sucessores) para o nó N6.

2-27 No Chord, considere que o sucessor do nó N12 é N17. Diga se o nó N12 é o predecessor de alguma das seguintes chaves:

a. c12 **b.** c15 **c.** c17 **d.** c22

2-28 Em uma rede Chord usando DHT com $m = 4$, desenhe o espaço de identificadores e insira 4 *peers* com IDs de nós nos endereços N3, N8, N11 e N13, bem como três chaves com endereços c5, c9 e c14. Determine qual nó é responsável por cada chave. Crie uma tabela de derivação para cada nó.

2-29 Em uma rede Chord usando DHT com $m = 4$, o nó N2 tem os seguintes valores na sua tabela de derivação: N4, N7, N10 e N12. Para cada uma das seguintes chaves, primeiro diga se N2 é o predecessor da chave. Se a resposta for não, encontre qual nó (o mais próximo predecessor) deve ser contatado para ajudar N2 a encontrar o predecessor de N2.

a. c1 **b.** c6 **c.** c9 **d.** c13.

2-30 Repita o Exemplo 2.16 apresentado no texto, mas desta vez considere que o nó N12 deseja encontrar o nó responsável pela chave c7. Dica: lembre-se de que a verificação de intervalo precisa ser feita usando aritmética módulo 32.

2-31 Repita o Exemplo 2.16 apresentado no texto, mas desta vez considere que o nó N5 deseja encontrar o nó responsável pela chave c16. Dica: lembre-se de que a verificação de intervalo precisa ser feita usando aritmética módulo 32.

2-32 No Pastry, considere que o espaço de endereços é 16 e que $b = 2$. Quantos dígitos aparecem em um endereço nesse espaço? Liste alguns dos identificadores.

2-33 Em uma rede Pastry com $m = 32$ e $b = 4$, qual é o tamanho da tabela de roteamento e do conjunto de folhas?

2-34 Mostre a estrutura básica de uma tabela de roteamento do Pastry para um espaço de endereços de 16 e tendo $b = 2$. Forneça algumas entradas possíveis para cada célula na tabela de roteamento do nó N21.

2-35 Em uma rede Pastry usando DHT na qual $m = 4$ e $b = 2$, desenhe o espaço de identificadores para quatro nós, N02, N11, N20 e N23, e três chaves, c00, c12 e c24. Determine qual nó é responsável por cada chave. Mostre também o conjunto de folhas e a tabela de roteamento para cada nó. Embora isso não seja realista, considere que a métrica de proximidade entre cada par de nós seja baseada na proximidade numérica.

2-36 Considerando o problema anterior, resolva as seguintes questões:

a. Mostre como o nó N02 responde a uma consulta em busca do nó responsável por c24.

b. Mostre como o nó N20 responde a uma consulta em busca do nó responsável por c12.

2-37 Usando a árvore binária da Figura 2.55 apresentada no texto, mostre a subárvore para o nó N11.

2-38 Usando as tabelas de roteamento da Figura 2.56 apresentada no texto, explique e mostre a rota se o nó N0 receber uma mensagem de busca tendo como alvo o nó responsável por c12.

2-39 Em uma rede Kademlia com $m = 4$, temos cinco nós ativos: N2, N3, N7, N10 e N12. Encontre a tabela de roteamento para cada nó ativo (com apenas uma coluna).

2-40 Na Figura 2.60 apresentada no texto, como o servidor sabe que um cliente solicitou um serviço?

2-41 Na Figura 2.62 apresentada no texto, como o *socket* é criado para transferir dados no lado do servidor?

2-42 Suponha que queremos fazer com que o programa-cliente TCP apresentado na Tabela 2.25 seja mais genérico, capaz de enviar uma sequência de caracteres e tratar a resposta recebida do servidor. Mostre como isso pode ser feito.

2.8 EXPERIMENTOS DE SIMULAÇÃO

2.8.1 *Applets*

Criamos alguns *applets* Java para demonstrar alguns dos principais conceitos discutidos neste capítulo. Recomendamos que o estudante ative esses *applets* no *site* www.mhhe.com/forouzan e examine cuidadosamente os protocolos em ação.

2.8.2 Experimentos de laboratório

No Capítulo 1, obtivemos e instalamos o Wireshark e aprendemos suas funcionalidades básicas. Neste capítulo, usamos o Wireshark para capturar e analisar alguns protocolos da camada de aplicação. Usamos o Wireshark para simular seis protocolos: HTTP, FTP, TELNET, SMTP, POP3 e DNS.

Lab2-1 No primeiro experimento, obtivemos páginas Web usando HTTP. Usamos o Wireshark para capturar pacotes para a análise. Aprendemos sobre as mensagens HTTP mais comuns. Também capturamos e analisamos as mensagens de resposta. Durante esta sessão de laboratório, alguns cabeçalhos HTTP também serão examinados e analisados.

Lab2-2 No segundo experimento, usamos o FTP para transferir alguns arquivos. Usamos o Wireshark para capturar alguns pacotes. Mostramos que o FTP usa duas conexões separadas: uma de controle e uma de transferência de dados. A conexão de dados é aberta e então fechada para cada atividade de transferência de arquivo. Mostramos também que o FTP é um protocolo de transferência de arquivos inseguro, porque as transações são feitas sem o uso de técnicas de criptografia.

Lab2-3 No terceiro experimento, usamos o Wireshark para capturar pacotes trocados pelo protocolo TELNET. Como no FTP, podemos observar nos pacotes capturados os comandos e as respostas trocados durante a sessão. Como o FTP, o TELNET é vulnerável a ataques, pois envia todos os dados, incluindo a senha, às claras.

Lab2-4 No quarto experimento, investigamos o protocolo SMTP em ação. Enviamos um e-mail e, usando o Wireshark, analisamos o conteúdo e o formato do pacote SMTP trocado entre o cliente e o servidor. Verificamos que as três fases discutidas ao longo do texto realmente aparecem nesta sessão SMTP.

Lab2-5 No quinto experimento, investigamos os estados e o comportamento do protocolo POP3. Recuperamos os e-mails armazenados em nossa caixa postal em um servidor POP3 e observamos e analisamos os estados do POP3, bem como o tipo e o conteúdo das mensagens trocadas, por meio da análise de pacotes capturados pelo Wireshark.

Lab2-6 No sexto experimento, analisamos o comportamento do protocolo DNS. Além do Wireshark, existem diversos utilitários de rede capazes de encontrar algumas informações armazenadas nos servidores DNS. Neste laboratório, usamos os utilitários *dig* (que substituiu o *nslookup* no UNIX e Linux). Também usamos o *ipconfig* para gerenciar os registros DNS armazenados no *cache* do *host*. Quando usamos esses utilitários, configuramos o Wireshark para capturar os pacotes enviados por eles.

2.9 TAREFAS DE PROGRAMAÇÃO

Escreva, compile e teste os programas a seguir utilizando a linguagem de programação de sua preferência:

Prg2-1 Escreva um programa para tornar o programa-servidor UDP da Tabela 2.22 mais genérico: ele deve ser capaz de receber uma requisição, processar a requisição e enviar de volta a resposta.

Prg2-2 Escreva um programa para tornar o programa-servidor UDP da Tabela 2.23 mais genérico: ele deve ser capaz de enviar qualquer pedido criado pelo programa-cliente.

Prg2-3 Escreva um programa para tornar o programa-servidor TCP da Tabela 2.24 mais genérico: ele deve ser capaz de receber uma requisição, processar a requisição e enviar de volta a resposta.

CAPÍTULO

3 CAMADA DE TRANSPORTE

A camada de transporte está localizada entre a camada de aplicação e a camada de rede na pilha de protocolos TCP/IP. Ela fornece serviços à camada de aplicação e recebe serviços da camada de rede. A camada de transporte atua como uma ligação entre um programa-cliente e um programa-servidor, como uma conexão de processo para processo. A camada de transporte é o coração da pilha de protocolos TCP/IP; ela é o meio lógico fim a fim para transferir dados de um ponto a outro na Internet. Este capítulo é um pouco longo porque precisamos abordar diversos assuntos e introduzir alguns conceitos novos. Ele está dividido em quatro seções:

- Na primeira seção, apresentamos os serviços gerais normalmente requisitados da camada de transporte, como comunicação processo a processo, endereçamento, multiplexação e demultiplexação, e controle de erros, de fluxo e de congestionamento.
- Na segunda seção, discutimos protocolos comuns da camada de transporte, como o *Stop-and-Wait*, *Go-Back-N* e Repetição Seletiva. Esses protocolos concentram-se nos serviços de controle de erros e de fluxo fornecidos por um protocolo real da camada de transporte. Entender esses protocolos nos ajuda a compreender melhor o projeto dos protocolos TCP e UDP e as razões por trás dos seus modelos.
- Na terceira seção, nos concentramos no UDP, que é o mais simples dos dois protocolos que discutimos neste capítulo. O UDP carece de muitos serviços que podem ser necessários em um protocolo da camada de transporte, mas a sua simplicidade é muito atrativa para algumas aplicações, conforme mostraremos.
- Na quarta seção, discutimos o TCP. Primeiro listamos seus serviços e suas características e, depois, mostramos como o TCP fornece um serviço orientado à conexão usando um diagrama de transição. Por fim, mostramos o controle de erros e de fluxo, usando janelas abstratas empregadas pelo TCP. O controle de congestionamento no TCP é discutido em seguida, assunto que será revisto no próximo capítulo para a camada de rede. Também fornecemos uma lista de temporizadores TCP e de suas funções, os quais são utilizados em diferentes serviços fornecidos pelo TCP, que também usa algumas opções disponíveis para consulta no *site* do livro para estudo posterior.

3.1 INTRODUÇÃO

A camada de transporte está localizada entre a camada de aplicação e a camada de rede. Ela fornece comunicação processo a processo entre duas camadas de aplicação, uma na estação local e outra na remota. A comunicação é fornecida por meio de uma conexão lógica, isto é, as duas camadas de aplicação, que podem estar localizadas em diferentes partes do mundo, assumem a existência de uma conexão imaginária direta através da qual elas podem enviar e receber mensagens. A Figura 3.1 mostra a ideia por trás de tal conexão lógica.

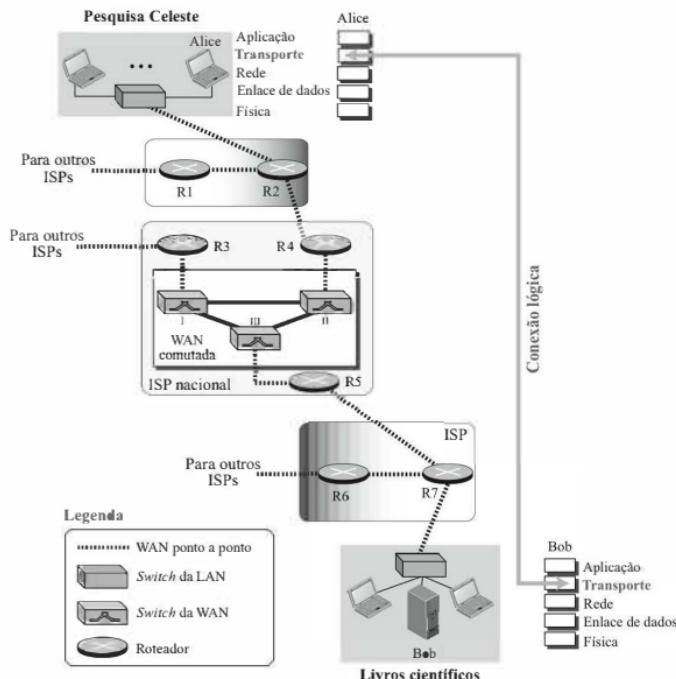



Figura 3.1 Conexão lógica na camada de transporte.

A figura mostra o mesmo cenário utilizado no Capítulo 2 para a camada de aplicação (Figura 2.1). A estação de Alice na empresa Pesquisa Celeste cria uma conexão lógica na camada de transporte com a estação de Bob na empresa Livros Científicos. As duas empresas se comunicam na camada de transporte, como se houvesse uma conexão real entre elas. A Figura 3.1 mostra que apenas os dois sistemas finais (computadores de Alice e Bob) usam o serviço da camada de transporte, enquanto todos os roteadores intermediários utilizam apenas as três primeiras camadas.

3.1.1 Serviços da camada de transporte

Conforme discutimos no Capítulo 1, a camada de transporte fica localizada entre a camada de rede e a camada de aplicação. A camada de transporte é responsável por prestar serviços à camada de aplicação; ela recebe serviços da camada de rede. Nesta seção, discutimos os serviços que podem ser fornecidos pela camada de transporte; na próxima seção, discutiremos vários protocolos da camada de transporte.

Comunicação processo a processo

O primeiro dever de um protocolo de camada de transporte é proporcionar **comunicação processo a processo**. Um processo é uma entidade da camada de aplicação (programa em execução) que usa os serviços da camada de transporte. Antes de discutirmos como a comunicação processo a processo pode ser feita, precisamos entender a diferença entre a comunicação *host* a *host* e comunicação processo a processo.

A camada de rede (discutida no Capítulo 4) é responsável pela comunicação no nível dos computadores (comunicação *host* a *host*). Um protocolo de camada de rede pode entregar a mensagem apenas para o computador de destino, mas a entrega ainda não está completa. A mensagem ainda precisa ser entregue ao processo correto. Esse é o ponto no qual um protocolo da camada de transporte assume a entrega. Um protocolo de camada de transporte é responsável pela entrega da mensagem para o processo apropriado. A Figura 3.2 mostra os domínios de uma camada de rede e de uma camada de transporte.

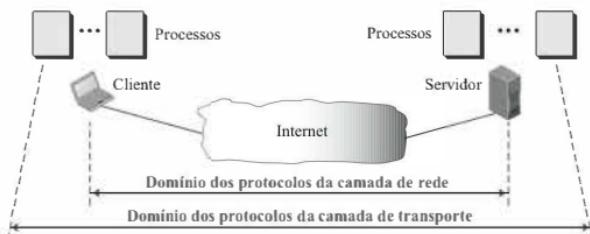



Figura 3.2 Camada de rede *versus* camada de transporte.

Endereçamento: números de porta

Embora existam algumas maneiras para se obter uma comunicação processo a processo, o mais comum é por meio do **paradigma cliente-servidor** (ver Capítulo 2). Um processo na estação local, denominado um *cliente*, necessita de serviços oferecidos por um processo normalmente localizado em uma estação remota, denominado *servidor*.

Ambos os processos (cliente e servidor) têm o mesmo nome. Por exemplo, para obter o dia e a hora de uma máquina remota, precisamos de um processo-cliente de *daytime* (hora local) sendo executado na estação local e também de um processo-servidor de *daytime* sendo executado na máquina remota.

No entanto, os sistemas operacionais atuais suportam tanto ambientes multiusuário como de multiprogramação. Um computador remoto pode executar vários programas-servidor ao mesmo tempo, assim como vários computadores locais podem executar um ou mais programas-cliente ao mesmo tempo. Na comunicação, precisamos definir a estação local, o processo local, a estação remota e o processo remoto. A estação local e a remota são definidas usando os endereços IP (discutidos no Capítulo 4). Para definir os processos, precisamos de um segundo grupo de identificadores, chamados **números de porta**. Na pilha de protocolos TCP/IP, os números de porta são números inteiros entre 0 e 65.535 (16 *bits*).

O programa-cliente define a si mesmo com um número de porta, chamado **número de porta efêmero**. O termo efêmero significa *de curta duração* e é utilizado porque o tempo de vida de um cliente é normalmente curto. Recomenda-se que um número de porta efêmero seja maior que 1.023 para que alguns programas cliente/servidor funcionem corretamente.

O processo-servidor também precisa definir a si mesmo com um número de porta, que, no entanto, não pode ser escolhido aleatoriamente. Se o computador no lado do servidor executa um processo-servidor e atribui um número aleatório como o número de porta, o processo no lado do cliente que deseja acessar esse servidor e utilizar os seus serviços não conhecerá tal número da porta. Naturalmente, uma solução seria enviar um pacote especial e solicitar o número da porta de um processo-servidor específico, mas isso criaria maior complexidade. O TCP/IP decidiu usar números de porta universais para os processos-servidor; estes são chamados **números de porta bem conhecidos**. Há algumas exceções a essa regra; por exemplo, há processos-cliente aos quais são atribuídos números de porta bem conhecidos. Todo processo-cliente sabe o número de porta bem conhecido do processo-servidor correspondente. Por exemplo, enquanto o processo-cliente de *daytime*, discutido anteriormente, pode usar um número de porta efêmero (temporário) de 52.000 para se identificar, o processo-servidor de *daytime* deve usar o número de porta bem conhecido (permanente) 13. A Figura 3.3 mostra esse conceito.

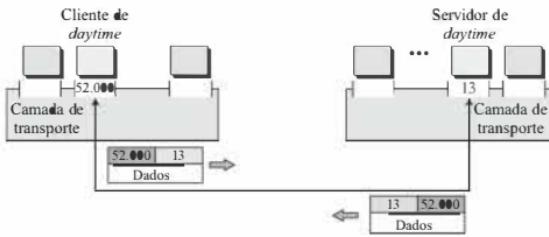



Figura 3.3 Números de porta.

Deve estar claro agora que os endereços IP e os números de porta desempenham papéis diferentes na escolha do destino final dos dados. O endereço IP de destino define a estação entre as diferentes máquinas espalhadas pelo mundo. Depois que a estação tiver sido selecionada, o número da porta define um dos processos nessa estação em particular (ver Figura 3.4).

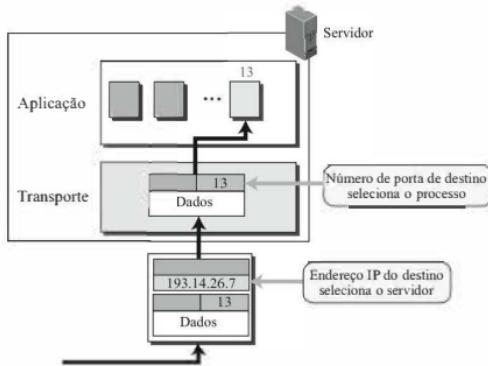



Figura 3.4 Endereços IP *versus* números de porta.

Faixas ICANN

A ICANN (ver Apêndice D no *site* www.grupoa.com.br) dividiu os números de porta em três faixas: bem conhecidos, registrados e dinâmicos (ou privados), conforme mostra a Figura 3.5.

Figura 3.5 Faixas de endereços ICANN.

- **Portas bem conhecidas.** As portas na faixa de 0 a 1.023 são atribuídas e controladas pela ICANN. São as portas bem conhecidas.
- **Portas registradas.** As portas na faixa de 1.024 a 49.151 não são atribuídas e controladas pela ICANN. Só podem ser registradas junto à ICANN para evitar duplicações.
- **Portas dinâmicas.** As portas na faixa de 49.152 a 65.535 não são controladas nem registradas. Esses números podem ser utilizados para portas temporárias ou privadas.

Exemplo 3.1

Nos sistemas UNIX, as portas bem conhecidas são armazenadas em um arquivo chamado /etc/services. Cada linha desse arquivo fornece o nome do servidor e o número de porta bem conhecido associado a ele. Podemos usar o utilitário *grep* para extrair a linha correspondente de uma aplicação desejada. A seguir, mostramos a porta para o TFTP. Perceba que o TFTP pode usar a porta 69 do UDP ou do TCP.

```
$grep tftp/etc/services
tftp 69/tcp
tftp 69/udp
```

O SNMP (ver Capítulo 9) usa dois números de porta (161 e 162), cada um para uma finalidade diferente.

```
$grep snmp/etc/services
snmp161/tcp#Simple Net Mgmt Proto
snmp161/udp#Simple Net Mgmt Proto
snmptrap162/udp#Traps for SNMP
```

Endereços de *socket*

Um protocolo da camada de transporte da pilha de protocolos TCP/IP requer um endereço IP e um número de porta, em cada extremidade, para estabelecer uma conexão. A combinação de um endereço IP e de um número de porta é chamada **endereço de *socket*.** O endereço de *socket* do cliente define o processo-cliente univocamente, assim como o endereço de *socket* do servidor define um processo-servidor univocamente (ver Figura 3.6).

Para utilizar os serviços da camada de transporte na Internet, precisamos de um par de endereços de *socket*: o endereço de *socket* cliente e o endereço de *socket* servidor. Essas quatro informações fazem parte do cabeçalho do pacote da camada de rede e do cabeçalho do pacote da camada de transporte. O primeiro cabeçalho contém os endereços IP, enquanto o segundo, os números de porta.

Figura 3.6 Endereço de *socket*.

Encapsulamento e desencapsulamento

Para enviar uma mensagem de um processo para outro, o protocolo da camada de transporte encapsula e desencapsula mensagens (Figura 3.7). O encapsulamento acontece no lado do remetente. Quando um processo possui uma mensagem para enviar, ele passa a mensagem para a camada de transporte juntamente com um par de endereços de *socket* e algumas outras informações que dependem do protocolo da camada de transporte. Esta recebe os dados e adiciona o cabeçalho da camada de transporte. Os pacotes nas camadas de transporte na Internet são denominados *datagramas de usuário*, *segmentos ou pacotes*, dependendo do protocolo que utilizamos. De uma maneira geral, nos referimos à carga útil da camada de transporte como pacotes.

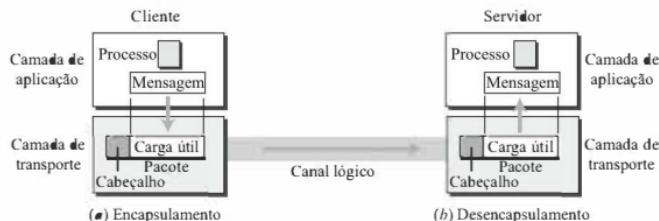



Figura 3.7 Encapsulamento e desencapsulamento.

O desencapsulamento acontece no lado do receptor. Quando a mensagem chega à camada de transporte do destino, o cabeçalho é descartado e a camada de transporte envia a mensagem para o processo em execução na camada de aplicação. O endereço de *socket* do remetente é passado para o processo caso ele precise responder à mensagem recebida.

Multiplexação e demultiplexação

Sempre que uma entidade aceita itens de mais de uma origem, temos algo denominado **multiplexação** (muitos para um); sempre que uma entidade encaminha itens a mais de uma origem, temos algo denominado **demultiplexação** (um para muitos). A camada de transporte na origem realiza a multiplexação, a camada de transporte no destino executa a demultiplexação (Figura 3.8).

A Figura 3.8 mostra a comunicação entre um cliente e dois servidores. Três processos-cliente estão sendo executados no lado do cliente, P1, P2 e P3. Os processos P1 e P3 precisam enviar solicitações para o processo-servidor correspondente, que está sendo executado em um servidor. O processo-cliente P2 precisa enviar uma solicitação para o processo-servidor correspondente que está em execução em outro servidor. A camada de transporte no lado do cliente aceita as três mensagens dos três processos e cria três pacotes. Ela atua como um *multiplexador*. Os pacotes 1 e 3 utilizam o mesmo canal lógico para atingir a camada de transporte do primeiro servidor. Quando eles chegam ao servidor, a camada de transporte faz

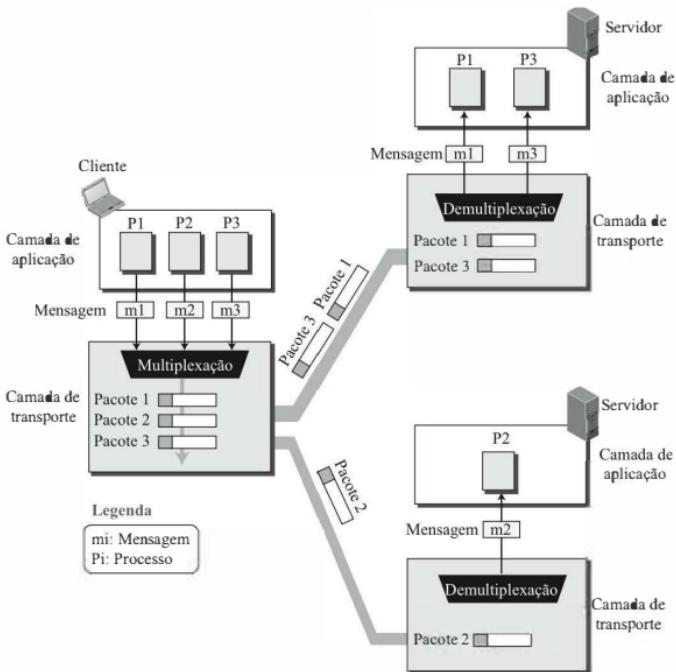



Figura 3.8 Multiplexação e demultiplexação.

o trabalho de um ***demultiplexador*** e distribui as mensagens para dois processos diferentes. A camada de transporte no segundo servidor recebe o pacote 2 e o entrega ao processo correspondente. Perceba que ainda temos demultiplexação, embora exista apenas uma mensagem.

Controle de fluxo

Sempre que uma entidade produz dados e outra entidade os consome, deve haver um equilíbrio entre as taxa de produção e de consumo. Se os dados são produzidos mais rapidamente do que podem ser consumidos, o consumidor pode ficar sobrecarregado e precisar descartar alguns dados. Se os dados são produzidos mais lentamente do que podem ser consumidos, o consumidor precisa esperar e o sistema torna-se menos eficiente. O controle de fluxo está relacionado com a primeira questão. Precisamos evitar a perda de dados no lado do consumidor.

Pushing ou *pulling*

A entrega dos dados de um produtor para um consumidor pode ocorrer de duas maneiras: *pushing* ("empurrando") ou *pulling* ("puxando"). Se o remetente envia os dados assim que são produzidos,

sem antes receber do consumidor um pedido de envio, a entrega é defina como *pushing*. Se o produtor envia os dados após o consumidor os ter solicitado, a entrega é definida como *pulling*. A Figura 3.9 mostra esses dois tipos de entrega.

Figura 3.9 *Pushing* ou *pulling*.

Quando o produtor "empurra" os dados, o consumidor pode ficar sobrecarregado e há uma necessidade de controle de fluxo, na direção oposta, para evitar o descarte de dados. Em outras palavras, o consumidor deve advertir o produtor para que este suspenda o envio e para informá-lo sobre quando estará novamente apto a receber dados. Quando o consumidor "puxa" os dados, ele solicitará mais dados assim que estiver pronto. Nesse caso, não há necessidade de controle de fluxo.

Controle de fluxo na camada de transporte

Na comunicação na camada de transporte, estamos lidando com quatro entidades: o processo emissor, a camada de transporte do emissor, a camada de transporte do receptor e o processo receptor. O processo emissor na camada de aplicação é apenas um produtor. Ele produz blocos de mensagens e os entrega ("empurrando") à camada de transporte. A camada de transporte do emissor tem um papel duplo: ela é tanto um consumidor como um produtor. Ela consome as mensagens "empurradas" pelo produtor, encapsula as mensagens em pacotes e as encaminha ("empurra") para a camada de transporte do receptor. Esta também tem um papel duplo, já que é o consumidor dos pacotes recebidos (enviados pelo emissor) e o produtor que desencapsula as mensagens e as entrega à camada de aplicação. A última entrega, no entanto, é normalmente uma entrega do tipo *pulling*; a camada de transporte espera até que o processo da camada de aplicação solicite mais mensagens.

A Figura 3.10 mostra que precisamos de controle de fluxo em pelo menos dois casos: entre a camada de transporte do emissor e a camada de aplicação do emissor e entre a camada de transporte do receptor e a camada de transporte do emissor.

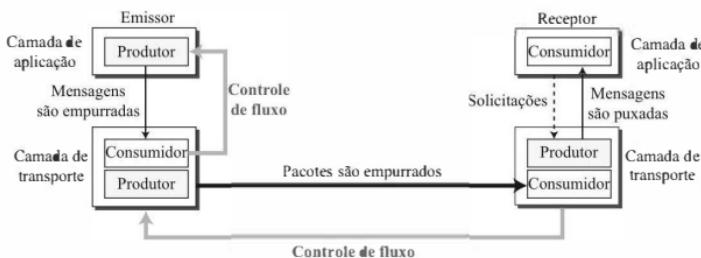



Figura 3.10 Controle de fluxo na camada de transporte.

Unidades de armazenamento temporário

Embora o controle de fluxo possa ser implementado de várias maneiras, uma das soluções mais comuns é o uso de duas unidades de armazenamento temporário (conhecidas como *buffers*): uma na camada de transporte emissora e outro na receptora. Um *buffer* é um conjunto de posições de memória que podem armazenar os pacotes no emissor e no receptor. A comunicação relativa ao controle de fluxo pode ocorrer por meio do envio de sinais do consumidor para o produtor.

Quando o *buffer* na camada de transporte emissora está cheio, ele informa a camada de aplicação para que ela suspenda a entrega de mensagens. Quando há algum espaço vago no *buffer*, ele informa a camada de aplicação que ela pode novamente enviar mensagens.

Quando o *buffer* da camada de transporte receptora está cheio, ele informa a camada de transporte emissora para que ela suspenda o envio de pacotes. Quando há espaço vago, ele informa que ela pode novamente enviar pacotes.

Exemplo 3.2

A discussão anterior requer que os consumidores se comuniquem com os produtores em duas ocasiões: quando o *buffer* está cheio e quando há espaço vago. Se as duas partes usarem um *buffer* com apenas uma posição, a comunicação pode ser facilitada. Considere que cada camada de transporte utilize um único endereço de memória para armazenar um pacote. Quando essa única posição na camada de transporte emissora estiver vazia, esta notifica a camada de aplicação para que ela envie a próxima mensagem; quando essa única posição na camada de transporte receptora estiver vazia, é enviada uma confirmação para a camada de transporte emissora, que pode então enviar o seu próximo pacote. Conforme veremos adiante, no entanto, esse tipo de controle de fluxo usando um *buffer* com uma única posição no emissor e no receptor é ineficiente.

Controle de erros

Na Internet, como a camada de rede subjacente (IP) não é confiável, precisamos tornar a camada de transporte confiável, caso a aplicação o exija. Essa confiabilidade pode ser alcançada por meio da adição de serviços de controle de erros na camada de transporte. O controle de erros na camada de transporte é responsável por

1. Detectar e descartar pacotes corrompidos.
2. Rastrear pacotes perdidos e descartados, bem como reenviá-los.
3. Reconhecer pacotes duplicados e descartá-los.
4. Manter pacotes fora da ordem no *buffer* até que os pacotes faltantes cheguem.

O controle de erros, ao contrário do de fluxo, envolve apenas as camadas de transporte emissora e receptora. Estamos considerando que as mensagens trocadas entre as camadas de aplicação e de transporte não apresentam erros. A Figura 3.11 mostra o controle de erros entre as camadas de transporte emissora e receptora. Tal como acontece no caso do controle de fluxo, a camada de transporte receptora gerencia o controle de erros, na maior parte do tempo, informando a camada de transporte emissora sobre problemas.

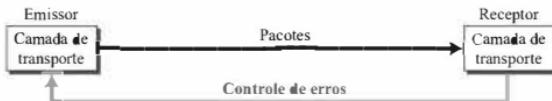



Figura 3.11 Controle de erros na camada de transporte.

Números de sequência

O controle de erros exige que a camada de transporte emissora saiba qual pacote deve ser reenviado e que a camada de transporte receptora saiba qual pacote foi duplicado e qual chegou fora de ordem. Isso pode ser feito se os pacotes forem numerados. Podemos acrescentar um campo no pacote da camada de transporte para carregar o **número de sequência** do pacote. Quando um pacote for corrompido ou perdido, a camada de transporte receptora pode informar de alguma maneira a camada de transporte emissora para que ela o reenvie, usando como base o número de sequência. A camada de transporte receptora pode também detectar a duplicação de pacotes se dois pacotes recebidos tiverem o mesmo número de sequência. Os pacotes fora de ordem podem ser reconhecidos observando-se lacunas entre os números de sequência.

Os pacotes são numerados sequencialmente. No entanto, como precisamos incluir o número de sequência de cada pacote no cabeçalho, devemos definir um limite. Se o cabeçalho do pacote suportar um número de sequência de m *bits*, os números de sequência podem variar de 0 a $2^m - 1$. Por exemplo, se m for 4, os únicos números de sequência possíveis ficam na faixa de 0 a 15, incluindo esses extremos. No entanto, podemos reiniciar o número de sequência ciclicamente. Assim, os números de sequência nesse caso são

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, ...

Em outras palavras, os números de sequência são módulo 2^m.

Para o controle de erros, os números de sequência são módulo 2^m, onde m é o tamanho do campo de número de sequência, em *bits*.

Confirmação

Podemos usar sinalizações positivas e negativas para o controle de erros, mas discutiremos apenas as sinalizações positivas, que são as mais comuns na camada de transporte. O lado receptor pode enviar uma confirmação ou *acknowledgment* (ACK) para cada conjunto de pacotes que chegar são e salvo. O receptor pode simplesmente descartar os pacotes corrompidos. O emissor pode detectar a perda de pacotes se ele usar um temporizador. Quando um pacote é enviado, o remetente inicia o temporizador. Se um ACK não chegar antes do temporizador expirar, o emissor reenvia o pacote. Os pacotes duplicados podem ser descartados silenciosamente pelo receptor. Os pacotes fora de ordem podem ser descartados (tratados como pacotes perdidos pelo emissor) ou armazenados até que os pacotes faltantes cheguem.

Combinação de controle de fluxo e de erros

Discutimos que o controle de fluxo requer a utilização de dois *buffers*, um no lado do emissor e outro no lado do receptor. Discutimos também que o controle de erros requer o uso de números de sequência e confirmação por ambas as partes. Esses dois requisitos podem ser combinados se usarmos dois *buffers* numerados, um no emissor e outro no receptor.

No emissor, quando um pacote é preparado para ser enviado, usamos o número da próxima posição livre no *buffer*, x, como o número de sequência do pacote. Quando o pacote é enviado, uma cópia é armazenada na memória na posição x, aguardando a confirmação da outra extremidade. Quando uma confirmação relacionada a um pacote enviado chega, o pacote é eliminado e a respectiva posição da memória fica livre.

No receptor, quando um pacote com o número de sequência y chega, ele é armazenado na posição de memória y até que a camada de aplicação esteja pronta para recebê-lo. Uma confirmação pode ser enviada para anunciar a recepção do pacote y.

Janela deslizante

Como os números de sequência utilizados são módulo 2^m, um círculo pode representar a sequência de números de 0 a $2^m - 1$ (Figura 3.12). O *buffer* é representado como um conjunto de fatias, chamado de **janela deslizante** (*sliding window*), que ocupa parte do círculo em qualquer momento. No lado do emissor, quando um pacote é enviado, a fatia correspondente é marcada. Quando todas as fatias estiverem marcadas significa que o *buffer* está cheio e não há como aceitar mensagens adicionais da camada de aplicação. Quando chega uma confirmação, a fatia correspondente é desmarcada. Se algumas fatias consecutivas a contar do início da janela forem desmarcadas, ela desliza sobre a faixa correspondente aos números de sequência para permitir o aparecimento de mais fatias livres no fim dela. A Figura 3.12 mostra a janela deslizante no emissor. Os números de sequência são módulo 16 ($m = 4$) e o tamanho da janela é 7. Perceba que a janela deslizante é apenas uma abstração: em uma situação real, são usadas variáveis de computador para armazenar os números de sequência do próximo pacote a ser enviado e do último pacote enviado.

(*a*) Quatro pacotes foram enviados.

(*b*) Cinco pacotes foram enviados.

(*c*) Sete pacotes foram enviados; a janela está cheia.

(*d*) O pacote 0 foi confirmado; a janela desliza.

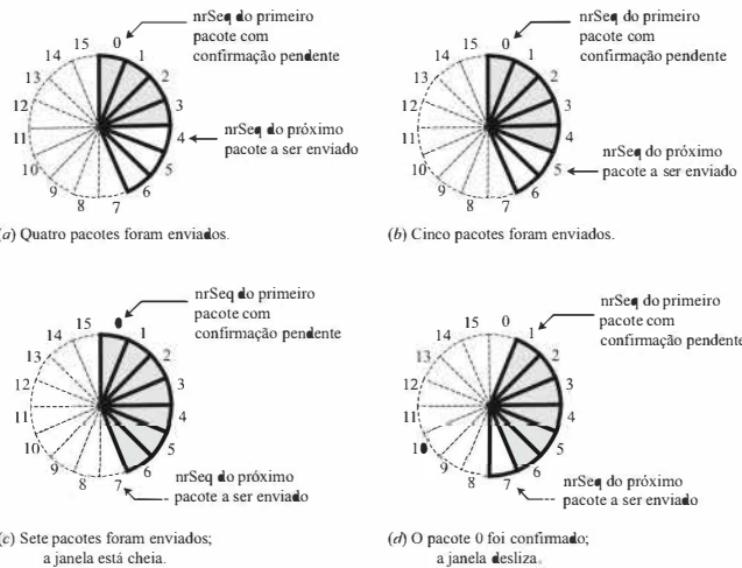



Figura 3.12 Janela deslizante em formato circular.

A maioria dos protocolos mostra a janela deslizante usando uma representação linear. A ideia é a mesma, mas normalmente ocupa menos espaço no papel. A Figura 3.13 mostra essa representação. Ambas as representações significam a mesma coisa. Se tomarmos os dois lados de cada diagrama da Figura 3.13 e curvá-los para cima, podemos criar o mesmo diagrama da Figura 3.12.

(*a*) Quatro pacotes foram enviados.

(*b*) Cinco pacotes foram enviados.

(*c*) Sete pacotes foram enviados; a janela está cheia.

(*d*) O pacote 0 foi confirmado; a janela desliza.

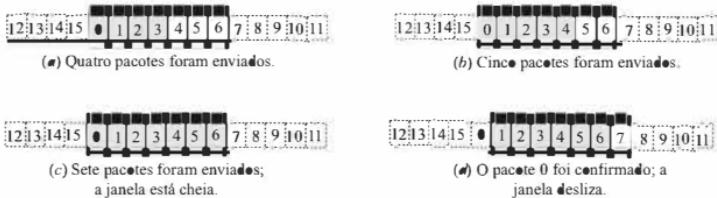



Figura 3.13 Janela deslizante em formato linear.

Controle de congestionamento

Uma questão importante em uma rede de comutação de pacotes, como a Internet, refere-se ao **congestionamento**. O congestionamento de uma rede pode ocorrer se a *carga* na rede – o número de pacotes enviados para ela – for maior que a *capacidade* da rede – o número de pacotes suportados por uma rede. O **controle de congestionamento** refere-se aos mecanismos e técnicas que controlam o congestionamento e mantêm a carga abaixo da capacidade.

Podemos perguntar por que existe congestionamento na rede. Ele ocorre em qualquer sistema que envolve espera. Por exemplo, o congestionamento acontece em uma rodovia, pois qualquer anormalidade no fluxo, como um acidente durante um horário de pico, cria um bloqueio.

O congestionamento em uma rede, ou conjunto de redes, ocorre porque os roteadores e *switches* possuem filas – *buffers* que guardam os pacotes antes e após o processamento. Um roteador, por exemplo, tem uma fila de entrada e uma de saída para cada interface. Se um roteador não for capaz de processar os pacotes com a mesma taxa na qual eles chegam, as filas ficam sobrecarregadas e ocorre um congestionamento. O congestionamento na camada de transporte é, na realidade, o resultado do congestionamento na camada de rede, que se manifesta na camada de transporte. Discutimos o congestionamento na camada de rede e suas causas no Capítulo 4. Mais adiante, mostramos como o TCP, assumindo que não há controle de congestionamento na camada de rede, implementa seu próprio mecanismo de controle de congestionamento.

Serviços orientados à conexão e não orientados à conexão

Um protocolo da camada de transporte, assim como um da camada de rede, pode fornecer dois tipos de serviços: orientado à conexão e não orientado à conexão. A natureza desses serviços na camada de transporte, no entanto, é diferente daquela dos serviços da camada de rede. Na camada de rede, um serviço não orientado à conexão pode significar caminhos distintos para diferentes datagramas pertencentes à mesma mensagem. Na camada de transporte, não estamos preocupados com os caminhos físicos dos pacotes (consideramos que exista uma conexão lógica entre as duas camadas de transporte). O serviço não orientado a conexão na camada de transporte significa a independência entre os pacotes; o serviço orientado à conexão significa que há uma dependência entre os pacotes. Discutiremos mais detalhadamente esses dois serviços.

Serviço não orientado à conexão

Em um serviço não orientado à conexão, o processo de origem (aplicativo) precisa dividir suas mensagens em blocos de dados com um tamanho que seja aceitável pela camada de transporte e entregá-los um a um para ela. A camada de transporte trata cada bloco como uma única unidade, sem qualquer relação entre os diferentes blocos. Quando chega um bloco proveniente da camada de aplicação, a camada de transporte o encapsula na forma de um pacote e o envia. Para ilustrar a independência entre os pacotes, considere que um processo-cliente possua três blocos de

mensagens para enviar a um processo-servidor. Os blocos são entregues ordenadamente a um protocolo de transporte não orientado à conexão. Entretanto, como não há uma dependência entre os pacotes na camada de transporte, eles podem chegar fora de ordem no destino e serem entregues fora de ordem para o processo-servidor (Figura 3.14).

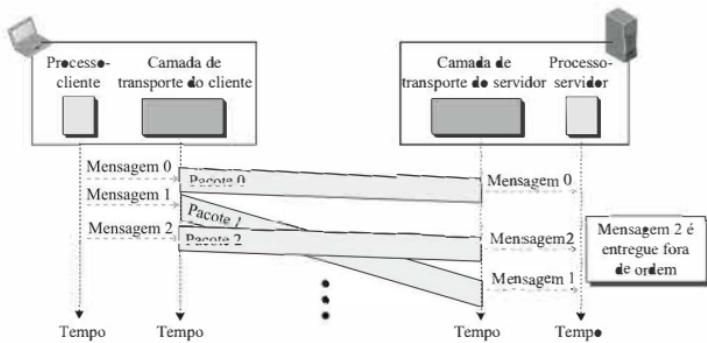



Figura 3.14 Serviço não orientado à conexão.

Na Figura 3.14, mostramos o movimento de pacotes usando uma linha de tempo, mas assumimos que as entregas dos dados pelo processo para a camada de transporte, e vice-versa, são instantâneas. A figura mostra que, no lado do cliente, os três blocos de mensagens são entregues ordenados à camada de transporte cliente (0, 1 e 2). Devido a um atraso extra no transporte do segundo pacote, a entrega das mensagens no servidor está fora de ordem (0, 2, 1). Se esses três blocos de dados pertencem à mesma mensagem, o processo-servidor pode receber uma mensagem estranha.

A situação poderia ser pior se um dos pacotes fosse perdido. Como não há numeração neles, a camada de transporte receptora não tem ideia de que uma das mensagens foi perdida. Ela apenas entrega dois blocos de dados para o processo-servidor.

Os dois problemas anteriores resultam do fato de que as duas camadas de transporte não são coordenadas entre si. A camada de transporte receptora não sabe quando o primeiro pacote chegará nem quando todos os pacotes chegaram.

Pode-se dizer que nenhum controle de fluxo, de erros ou de congestionamento pode ser efetivamente implementado em um serviço não orientado à conexão.

Serviço orientado à conexão

Em um serviço orientado à conexão, o cliente e o servidor precisam primeiro estabelecer uma conexão lógica entre eles. A troca de dados só pode acontecer após o estabelecimento da conexão. Após isso, ela precisa ser desfeita (Figura 3.15).

Conforme mencionamos anteriormente, o serviço orientado à conexão na camada de transporte é diferente do mesmo serviço na camada de rede. Na camada de rede, um serviço orientado à conexão significa que existe uma coordenação entre as duas estações finais e todos os roteadores entre elas. Na camada de transporte, o serviço orientado à conexão envolve apenas as duas estações; o serviço é fim a fim. Isto significa que devemos ser capazes de criar um protocolo orientado à conexão na camada de transporte sobre qualquer protocolo da camada de rede, seja ele orientado à conexão ou não. A Figura 3.15 mostra as fases de estabelecimento de conexão, transferência de dados e finalização de um serviço orientado à conexão na camada de transporte.

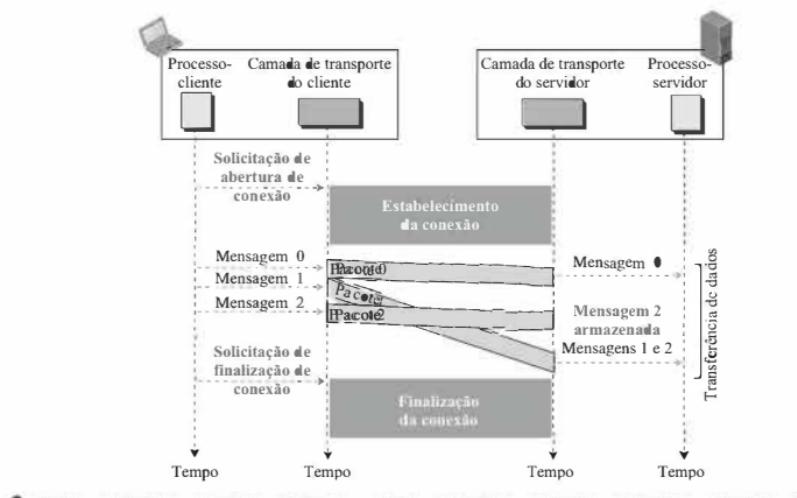



Figura 3.15 Serviço orientado à conexão.

Podemos implementar controle de fluxo, de erros e de congestionamento em um protocolo orientado à conexão.

Máquina de estados finitos

O comportamento de um protocolo da camada de transporte, tanto quando ele fornece um protocolo não orientado à conexão como quando fornece, pode ser melhor ilustrado como uma **Máquina de Estados Finitos** (MEF). A Figura 3.16 mostra uma representação de uma camada de transporte utilizando uma MEF. Usando essa ferramenta, cada camada de transporte (emissora ou receptora) é vista como uma máquina com um número finito de estados. Ela está sempre em um dos estados até que ocorra um evento; cada um está associado a duas reações: definição da lista (possivelmente vazia) de ações a serem executadas e determinação do próximo estado (que pode ser o próprio estado atual). Um dos estados deve ser definido como o inicial, o estado em que a máquina começa quando ela é ligada. Nessa figura, usamos retângulos com cantos arredondados para representar os estados, texto cinza para indicar os eventos e texto normal em preto para mostrar as ações. Uma linha horizontal é usada para separar o evento das ações, embora mais adiante substituamos essa linha horizontal por uma barra. Uma seta mostra a mudança para o próximo estado.

Podemos pensar em uma camada de transporte não orientada à conexão como uma MEF tendo um único estado: o estabelecido. A máquina em cada extremidade (cliente e servidor) está sempre no estado estabelecido, pronta para enviar e receber pacotes da camada de transporte.

Uma MEF em uma camada de transporte orientada à conexão, por outro lado, precisa passar por três estados antes de atingir o estado estabelecido. A máquina também precisa passar por três estados antes de fechar a conexão. A máquina está no estado *fechado* quando não há conexão. Ela mantém-se nesse estado até que um pedido de abertura de conexão chegue vindo do processo local; a máquina envia, então, um pacote de solicitação de abertura para a camada de transporte remota e vai para o estado *espera de abertura I*. Quando uma confirmação proveniente da outra extremidade

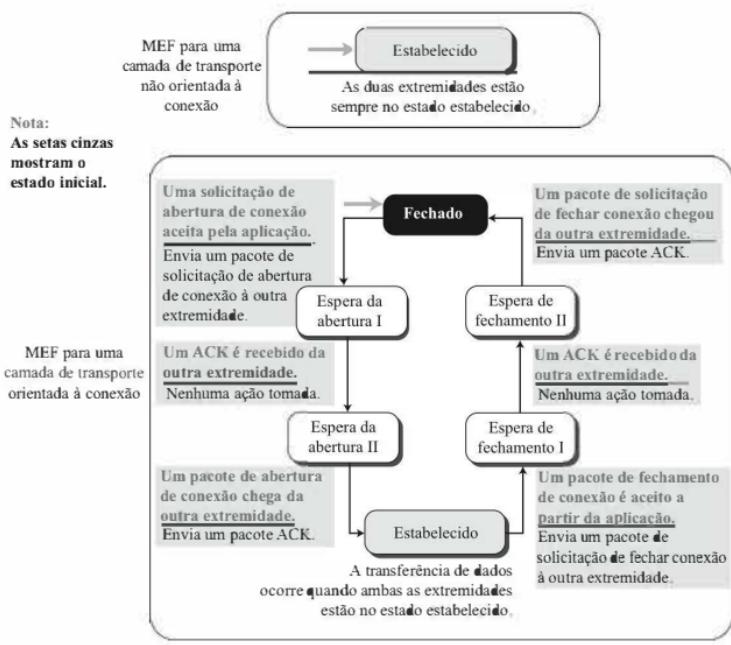



Figura 3.16 Serviços não orientados à conexão e orientados à conexão representados como MEFs.

é recebida, a MEF local muda para o estado *espera de abertura-II*. Quando a máquina está nesse estado, significa que uma conexão unidirecional foi estabelecida, mas se uma conexão bidirecional for necessária, a máquina precisa esperar nesse estado até que a outra extremidade também solicite uma conexão. Quando a solicitação for recebida, a máquina envia uma confirmação e muda para o estado *estabelecido*.

Dados e confirmações de dados podem ser trocados entre as duas extremidades quando ambas estiverem no estado estabelecido. No entanto, precisamos lembrar que o estado estabelecido, tanto nas camadas de transporte não orientadas à conexão como nas orientadas à conexão, representa um conjunto de estados de transferência de dados, assunto que discutiremos na próxima seção, sobre Protocolos da Camada de Transporte.

Para fechar uma conexão, a camada de aplicação envia uma mensagem de pedido de fechamento à sua camada de transporte local. Esta envia um pacote de solicitação para fechar conexão para a outra extremidade e muda o seu estado para *espera de fechamento I*. Quando uma confirmação é recebida da outra extremidade, a máquina muda para o estado *espera de fechamento II* e aguarda o pacote de solicitação para fechar conexão proveniente da outra extremidade. Quando esse pacote chega, a máquina envia uma confirmação e muda para o estado *fechado*.

Existem diversas variações da MEF orientada à conexão que discutiremos mais adiante. Também veremos como a MEF pode ser reduzida ou expandida e que os nomes dos estados podem ser alterados.

3.2 PROTOCOLOS DA CAMADA DE TRANSPORTE

Podemos criar um protocolo na camada de transporte por meio da combinação de um conjunto de serviços descritos nas seções anteriores. Para entender melhor o comportamento desses protocolos, começamos com o mais simples e gradualmente adicionamos mais complexidade. A pilha de protocolos TCP/IP utiliza um protocolo da camada de transporte que é uma modificação ou uma combinação de alguns deles. Discutimos esses protocolos genéricos nesta seção para abrir caminho para a compreensão dos protocolos mais complexos no restante do capítulo. Para simplificar a discussão, discutimos primeiramente todos esses protocolos como um protocolo unidirecional (ou seja, *simplex*) no qual os pacotes de dados se movem em uma única direção. No fim do capítulo, discutimos brevemente como eles podem ser alterados para protocolos bidirecionais nos quais os dados podem ser movidos nas duas direções (ou seja, *full-duplex*).

3.2.1 Protocolo simples

Nosso primeiro protocolo é simples, não orientado à conexão e sem controle de fluxo e de erros. Consideramos que o receptor possa lidar imediatamente com qualquer pacote que receba. Em outras palavras, o receptor não pode ser sobrecarregado com pacotes de entrada. A Figura 3.17 mostra a estrutura geral desse protocolo.

Figura 3.17 Protocolo simples.

A camada de transporte no lado do emissor recebe uma mensagem da sua camada de aplicação, cria um pacote a partir dela e então o envia. A camada de transporte no receptor recebe um pacote de sua camada de rede, extrai a mensagem dele, e a entrega à sua camada de aplicação. As camadas de transporte do emissor e do receptor fornecem o serviço de transmissão para as suas camadas de aplicação.

MEFs

O lado do emissor só deve enviar pacotes quando sua camada de aplicação tiver uma mensagem para enviar. O lado do receptor só pode entregar mensagens à sua camada de aplicação quando um pacote chegar. Podemos mostrar esses requisitos usando duas MEFs. Cada MEF tem apenas um estado, o *estado pronto*. A máquina emissora permanece no estado pronto até que uma solicitação chegue do processo da camada de aplicação. Quando isso ocorre, a máquina emissora encapsula a mensagem em um pacote e a envia à máquina receptora, que permanece no estado pronto até que um pacote chegue. Quando ocorre esse evento, a máquina receptora desencapsula a mensagem a partir do pacote e a entrega ao processo da camada de aplicação. A Figura 3.18 mostra as MEFs para esse protocolo simples. Veremos mais adiante que o protocolo UDP é uma ligeira modificação desse protocolo.

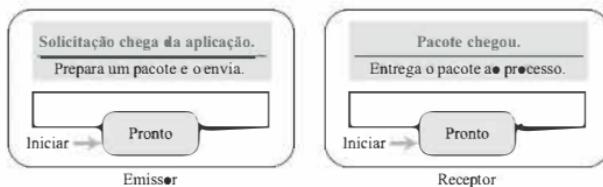



Figura 3.18 MEFs para um protocolo simples.

Exemplo 3.3

A Figura 3.19 mostra um exemplo de comunicação utilizando esse protocolo. Ele é muito simples. O emissor envia pacotes, um após o outro, sem sequer pensar sobre o receptor.

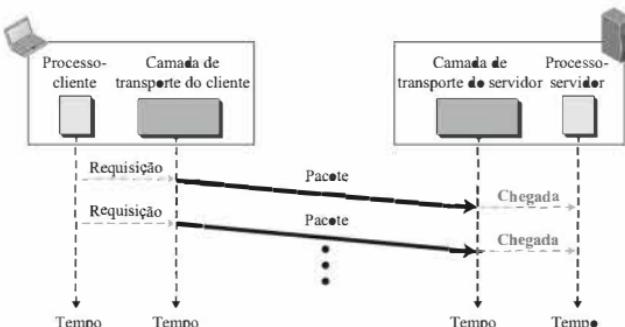



Figura 3.19 Diagrama de fluxo para o Exemplo 3.3.

3.2.2 Protocolo *Stop-and-Wait*

Nosso segundo protocolo é orientado à conexão e chamado **protocolo *Stop-and-Wait*** ou **protocolo Pare e Espere**, que utiliza tanto o controle de fluxo como o de erros. O emissor e o receptor usam ambos uma janela deslizante de tamanho 1. O emissor envia um pacote por vez e espera por uma confirmação antes de enviar o próximo. Para detectar pacotes corrompidos, precisamos adicionar uma soma de verificação (*checksum*) a cada pacote de dados. Quando um pacote chega no lado do receptor, ele é verificado. Se a sua soma de verificação estiver incorreta, o pacote está corrompido e é silenciosamente descartado. O silêncio do receptor é um sinal para o emissor de que um pacote foi corrompido ou perdido. Toda vez que o emissor envia um pacote, ele inicia um temporizador. Se a confirmação chegar antes que o temporizador expire, ele é interrompido e o emissor envia o próximo pacote (caso tenha algum para enviar). Se o temporizador expirar, o emissor reenvia o pacote anterior, considerando que o pacote foi perdido ou corrompido. Isto significa que o emissor precisa manter uma cópia do pacote até a chegada da confirmação (ACK) para ele. A Figura 3.20 mostra a estrutura geral do protocolo *Stop-and-Wait*. Perceba que apenas um pacote e uma confirmação podem estar nos canais ao mesmo tempo.

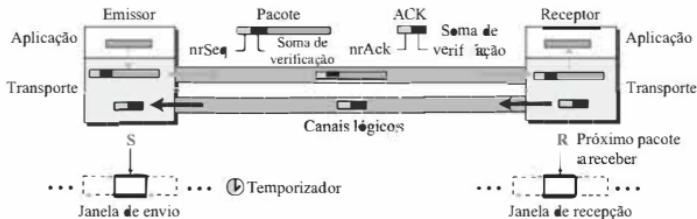



Figura 3.20 Protocolo *Stop-and-Wait*.

O protocolo *Stop-and-Wait* é um protocolo orientado à conexão que fornece controle de fluxo e de erros.

Números de sequência

Para evitar pacotes duplicados, o protocolo usa números de sequência e de confirmação. Um campo é adicionado ao cabeçalho do pacote para armazenar o número de sequência daquele pacote. Uma consideração importante refere-se à faixa dos números de sequência. Como queremos minimizar o tamanho do pacote, devemos usar o menor intervalo possível capaz de fornecer uma comunicação livre de ambiguidades. Discutiremos o intervalo de números de sequência necessário. Considere que temos x como um número de sequência; precisamos usar apenas $x + 1$ depois disso. Não há necessidade de $x + 2$. Para mostrar isso, considere que o emissor enviou um pacote com número de sequência x. Três coisas podem acontecer.

1. O pacote chega são e salvo no lado do receptor; este, então, envia uma confirmação, que chega no lado do emissor fazendo com que ele envie o próximo pacote numerado com $x + 1$.
2. O pacote foi corrompido ou nunca chegou ao receptor; o emissor reenvia o pacote (com a numeração x) após o tempo-limite ser atingido. O receptor retorna uma confirmação.
3. O pacote chega são e salvo no receptor; este, então, envia uma confirmação, mas ela é corrompida ou perdida. O emissor reenvia o pacote (com a numeração x) após o tempo-limite ser atingido. Perceba que esse pacote foi duplicado. O receptor é capaz de reconhecer esse fato, pois ele espera pelo pacote $x + 1$, mas o pacote recebido é o x.

Pode-se perceber que são necessários os números de sequência x e $x + 1$, pois o receptor precisa conseguir distinguir entre os casos 1 e 3. Contudo, não há necessidade de numerar um pacote com $x + 2$. No caso 1, o pacote pode ser numerado como x novamente, porque os pacotes x e $x + 1$ são confirmados e não há ambiguidade em nenhum dos lados. Nos casos 2 e 3, o novo pacote é numerado com $x + 1$, não com $x + 2$. Se apenas x e $x + 1$ são necessários, podemos usar a representação $x = 0$ e $x + 1 = 1$. Isto significa que a sequência é 0, 1, 0, 1, 0 e assim por diante; trata-se da aritmética módulo 2.

Números de confirmação

Como os números de sequência devem ser adequados tanto para pacotes de dados como para confirmações, usamos a seguinte convenção: os números de confirmação sempre anunciam o número de sequência do *próximo pacote esperado* pelo receptor. Por exemplo, se o pacote 0 chegou são e salvo, o receptor envia um ACK com o número de sequência igual a 1 (ou seja, o pacote 1 é o próximo pacote esperado). Se o pacote 1 chegou são e salvo, o receptor envia um ACK com o número de sequência igual a 0 (ou seja, o pacote 0 é o próximo pacote esperado).

No protocolo *Stop-and-Wait*, o número de confirmação sempre anuncia, em aritmética módulo 2, o número de sequência do próximo pacote esperado.

O emissor possui uma variável de controle, que denotamos por E (emissor), que aponta para a única posição da janela de envio. O receptor tem uma variável de controle, que denotamos por R (receptor), que aponta para a única posição da janela de recepção.

MEFs

A Figura 3.21 mostra as MEFs para o protocolo *Stop-and-Wait*. Como o protocolo é um protocolo orientado à conexão, ambas as extremidades devem estar no estado *estabelecido* antes de trocarem pacotes de dados. Os estados estão, na realidade, agrupados no estado *estabelecido*.

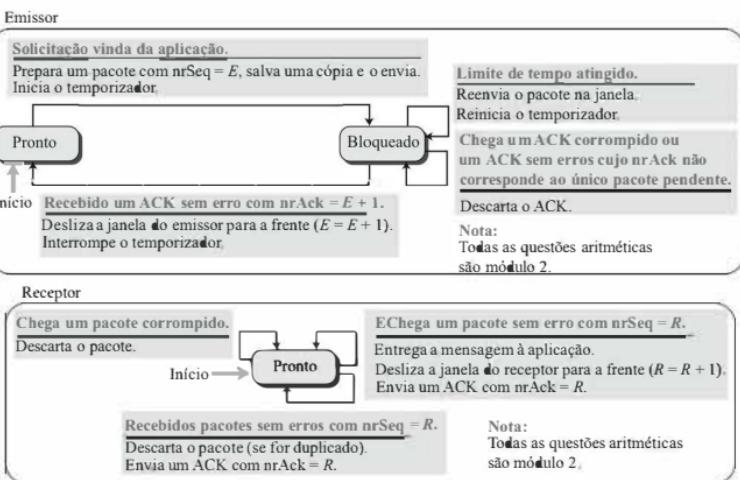



Figura 3.21 MEF para o protocolo *Stop-and-Wait*.

Emissor

O emissor está inicialmente no estado pronto, mas pode se mover entre os estados pronto e bloqueado. A variável E é inicializada com o valor 0.

- **Estado pronto.** Quando o emissor está nesse estado, ele está só esperando que um evento ocorra. Se chegar um pedido da camada de aplicação, o emissor cria um pacote com o número de sequência igual a E. Uma cópia do pacote é armazenada e o pacote é enviado. O emissor, em seguida, inicia o único temporizador, e passa, então, para o estado *bloqueado*.
- **Estado bloqueado.** Quando o emissor está nesse estado, três eventos podem ocorrer:
  - **a.** Se um ACK for recebido sem erros com o nrAck relativo ao próximo pacote a ser enviado, o que significa nrAck = $(E + 1)$ módulo 2, então o temporizador é parado. A janela desliza, $E = (E + 1)$ módulo 2. Finalmente, o emissor passa para o estado *pronto*.

b. Se for recebido um ACK corrompido ou um ACK sem erros, mas cujo nrAck ≠ $(E + 1)$ módulo 2, o ACK é descartado.

c. Se o tempo-limite for atingido, o emissor reenvia o pacote pendente e reinicia o temporizador.

Receptor

O receptor está sempre no estado *pronto*. Três eventos podem ocorrer:

a. Se um pacote sem erros com o nrSeq R for recebido, a mensagem do pacote é entregue à camada de aplicação. A janela desliza, $R = (R + 1)$ módulo 2. Finalmente, um ACK com nrAck = R é enviado.

b. Se um pacote sem erros com nrSeq ≠ R for recebido, o pacote é descartado, mas um ACK com nrAck = R é enviado.

c. Se um pacote corrompido for recebido, o pacote é descartado.

Exemplo 3.4

A Figura 3.22 mostra um exemplo do protocolo *Stop-and-Wait*. O pacote 0 é enviado e confirmado. O pacote 1 é perdido e reenviado após o tempo-limite ser atingido. O pacote 1 reenviado é confirmado e o temporizador é interrompido. O pacote 0 é enviado e confirmado, mas a confirmação se perde. O emissor não tem ideia se o pacote (ou a confirmação) foi perdido; então, após o tempo-limite ser atingido, ele reenvia o pacote 0, que é confirmado.

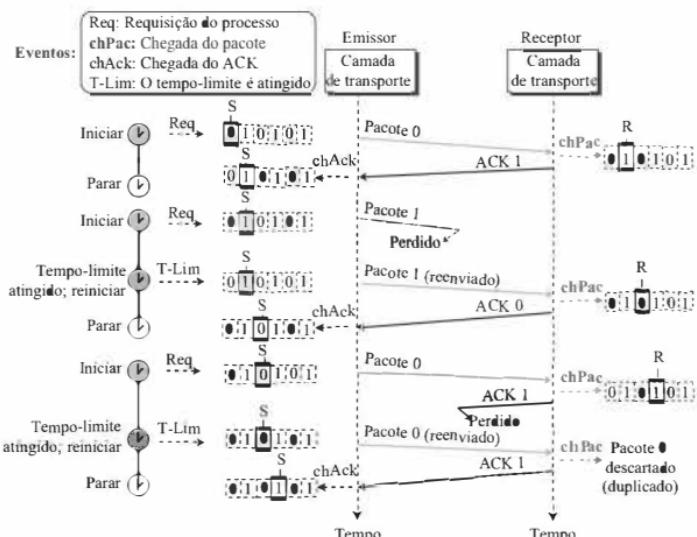



Figura 3.22 Diagrama de fluxo do Exemplo 3.4.

Eficiência

O protocolo *Stop-and-Wait* é muito ineficiente se o nosso canal for largo e comprido. Por largo, queremos dizer que o nosso canal tem uma ampla largura de banda (elevada taxa de transferência de dados); por comprido, queremos dizer que o atraso relativo ao tempo de ida e volta é longo. O produto dessas duas medidas é denominado **produto largura de banda-atraso**. Podemos imaginar esse canal como se fosse um tubo. O produto largura de banda-atraso corresponde ao volume do tubo em *bits*. O tubo está sempre lá e perde-se eficiência se não for usado. O produto largura de banda-atraso é a medida do número de *bits* que um emissor pode transmitir através do sistema enquanto espera por uma confirmação do receptor.

Exemplo 3.5

Considere que, em um sistema *Stop-and-Wait*, a largura de banda da linha seja de 1 Mbps e que 1 *bit* leve 20 milissegundos para fazer uma viagem de ida e volta. Qual é o produto largura de banda-atraso? Se os pacotes de dados do sistema tiverem um comprimento de 1.000 *bits*, qual será a porcentagem de utilização do enlace?

Solução

O produto largura de banda-atraso é $(1 \times 10^6) \times (20 \times 10^{-3}) = 20.000$ *bits*. O sistema pode enviar 20.000 *bits* durante o tempo que leva para os dados irem do emissor até o receptor e chegar a confirmação. Entretanto, o sistema envia apenas 1.000 *bits*. Podemos dizer que a utilização do enlace é de apenas 1.000 / 20.000, ou 5%. Por isso, em um enlace com uma largura de banda elevada ou atraso longo, a utilização do *Stop-and-Wait* desperdiça a capacidade do enlace.

Exemplo 3.6

Qual é a porcentagem de utilização do enlace no Exemplo 3.5, se tivermos um protocolo que pode enviar até 15 pacotes antes de parar e se preocupar com as confirmações?

Solução

O produto largura de banda-atraso ainda é de 20.000 *bits*. O sistema pode enviar até 15 pacotes ou 15.000 *bits* durante o tempo de ida e volta. Isto significa que a utilização é de 15.000 / 20.000, ou 75%. Obviamente, se houver pacotes corrompidos, a porcentagem de utilização é muito menor porque os pacotes precisam ser reenviados.

Encadeamento

Em redes e outras áreas, uma tarefa muitas vezes é iniciada antes que a anterior tenha terminado. Isto é conhecido como **encadeamento** (*pipelining*). Não há encadeamento no protocolo *Stop-and-Wait* porque o emissor deve esperar que o pacote chegue ao destino e seja confirmado antes que o próximo pacote possa ser enviado. No entanto, o encadeamento se aplica aos nossos próximos dois protocolos, porque vários pacotes podem ser enviados antes que o emissor receba qualquer notificação relativa aos pacotes anteriores. O encadeamento melhora a eficiência da transmissão, se o número de *bits* em trânsito for grande em relação ao produto largura de banda-atraso.

3.2.3 Protocolo *Go-Back-N*

Para melhorar a eficiência da transmissão (encher o tubo), múltiplos pacotes devem estar em trânsito enquanto o emissor espera a confirmação. Em outras palavras, precisamos deixar que mais de um pacote fique pendente para que sejamos capazes de manter o canal ocupado enquanto o emissor aguarda a confirmação. Nesta seção, discutimos um protocolo que pode atingir esse objetivo; na próxima seção, discutiremos um segundo protocolo. O primeiro protocolo é denominado ***Go-Back-N***

(GBN) ou ***Volte Atrás N*** (a razão para o nome ficará clara mais adiante). A chave para o *Go-back-N* é que podemos enviar diversos pacotes antes de receber as confirmações, mas o receptor só pode manter um pacote no *buffer*. Mantemos uma cópia dos pacotes enviados até que as confirmações cheguem. A Figura 3.23 mostra um esboço do protocolo. Observe que vários pacotes de dados e confirmações podem estar no canal ao mesmo tempo.

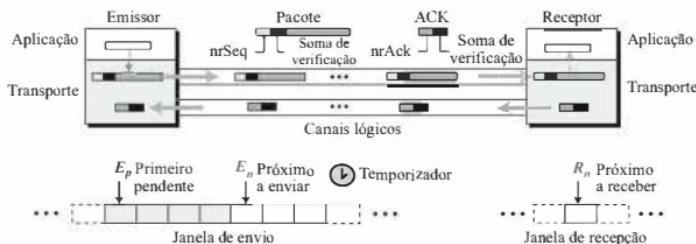



Figura 3.23 Protocolo *Go-Back-N*.

Números de sequência

Conforme mencionamos anteriormente, os números de sequência são módulo 2^m, onde m é o tamanho do campo de número de sequência, em *bits*.

Números de confirmação

Um número de confirmação neste protocolo é cumulativo e define o número de sequência do próximo pacote esperado. Por exemplo, se o número de confirmação (nrAck) for 7, isto significa que todos os pacotes com número de sequência até 6 chegaram sãos e salvos, de modo que o receptor está esperando o pacote com número de sequência igual a 7.

No protocolo *Go-Back-N*, o número de confirmação é cumulativo e define o número de sequência do próximo pacote que deve chegar.

Janela de envio

A janela de envio é uma espécie de caixa imaginária que cobre os números de sequência dos pacotes de dados que podem estar em trânsito ou podem ser enviados. Em cada posição da janela, algunsdesses números de sequência definem os pacotes que já foram enviados; outros definem os pacotes que podem ser enviados. O tamanho máximo da janela é de $2^m - 1$ por razões que discutiremos mais adiante. Neste capítulo, consideramos que o tamanho é fixo e tem o valor máximo, mas veremos mais adiante que alguns protocolos podem ter uma janela com tamanho variável. A Figura 3.24 mostra uma janela deslizante de tamanho 7 ($m = 3$) para o protocolo *Go-Back-N*.

A janela de envio pode, a qualquer momento, dividir os possíveis números de sequência em quatro regiões. A primeira região, à esquerda da janela, define os números de sequência pertencentes aos pacotes que já foram confirmados. O emissor não se preocupa com esses pacotes e não mantém cópias deles. A segunda região, na cor cinza, define o intervalo dos números de sequência

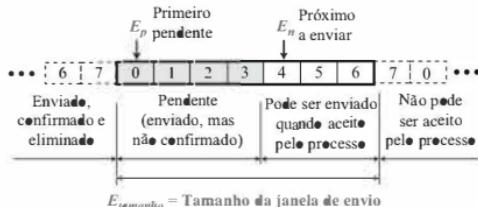



Figura 3.24 Janela de envio para o protocolo *Go-Back-N.*

pertencentes aos pacotes que já foram enviados, mas cujo estado é desconhecido. O emissor precisa esperar para descobrir se os pacotes foram recebidos ou perdidos. Eles são denominados pacotes *pendentes*. A terceira região, de cor branca na figura, define o intervalo dos números de sequência para os pacotes que podem ser enviados; no entanto, os dados correspondentes ainda não foram recebidos da camada de aplicação. Finalmente, a quarta região, à direita da janela, define os números de sequência que não podem ser usados até que a janela deslize.

A janela em si é uma abstração; três variáveis definem seu tamanho e sua localização em qualquer momento. Denotamos essas variáveis por E_p (na janela de envio, o primeiro pacote pendente), E_n (na janela de envio, o próximo pacote a ser enviado) e $E_{tamanho}$ (o tamanho da janela de envio). A variável E_p define o número de sequência do primeiro pacote pendente (o mais antigo). A variável E_n armazena o número de sequência que será atribuído ao próximo pacote a ser enviado. Finalmente, a variável $E_{tamanho}$ define o tamanho da janela, que é fixo no nosso protocolo.

A janela de envio é um conceito abstrato que define uma caixa imaginária com tamanho máximo = $2^m - 1$ e três variáveis: E_p, E_n e $E_{tamanho}$.

A Figura 3.25 ilustra como uma janela de envio pode deslizar uma ou mais posições para a direita quando uma confirmação chega da outra extremidade. Na figura, chega uma confirmação com nrAck = 6. Isto significa que o receptor está aguardando um pacote cujo número de sequência é 6.

Figura 3.25 Deslizando a janela de envio.

A janela de envio pode deslizar de uma ou mais posições quando chega um ACK sem erros cujo nrAck é maior do que ou igual a E_p e menor do que E_n (em aritmética modular).

Janela de recepção

A janela de recepção garante que os pacotes de dados corretos são recebidos e que as confirmações corretas são enviadas. No protocolo *Go-Back-N*, o tamanho da janela de recepção é sempre 1. O receptor está sempre esperando pela chegada de um pacote específico. Qualquer pacote que chegar fora de ordem é descartado e deve ser reenviado. A Figura 3.26 ilustra a janela de recepção. Perceba que precisamos de apenas uma variável, R_n (na janela de recepção, o próximo pacote esperado), para definir essa abstração. Os números de sequência no lado esquerdo da janela pertencem aos pacotes já recebidos e confirmados; os números de sequência à direita da janela definem os pacotes que não podem ser recebidos. Qualquer pacote recebido com um número de sequência nessas duas regiões é descartado. Apenas um pacote cujo número de sequência corresponde ao valor de R_n é aceito e confirmado. A janela de recepção também desliza, mas apenas uma posição de cada vez. Quando um pacote correto é recebido, a janela desliza $R_n = (R_n + 1)$ módulo 2^m.

Figura 3.26 Janela de recepção para o protocolo *Go-Back-N*.

A janela de recepção é um conceito abstrato que define uma caixa imaginária de tamanho 1, com uma única variável R_n. A janela desliza quando chega um pacote correto; o deslizamento é feito uma posição de cada vez.

Temporizador

Embora possa haver um temporizador para cada pacote enviado, no nosso protocolo, usamos um único temporizador. A razão é que o temporizador para o primeiro pacote pendente sempre atinge o tempo-limite primeiro. Reenviamos todos os pacotes pendentes quando o tempo-limite é atingido.

Reenviando pacotes

Quando o tempo-limite é atingido, o remetente reenvia todos os pacotes pendentes. Por exemplo, considere que o remetente já enviou o pacote 6 ($E_n = 7$), mas o tempo-limite no único temporizador é atingido. Se $E_p = 3$, isto significa que os pacotes 3, 4, 5 e 6 não foram confirmados; o emissor volta atrás e reenvia os pacotes 3, 4, 5 e 6. É por isso que o protocolo é chamado *Go-Back-N* (Volte Atrás *N*). Quando o tempo-limite é atingido, a máquina volta atrás *N* posições e reenvia todos os pacotes.

MEFs

A Figura 3.27 ilustra as MEFs para o protocolo GBN.

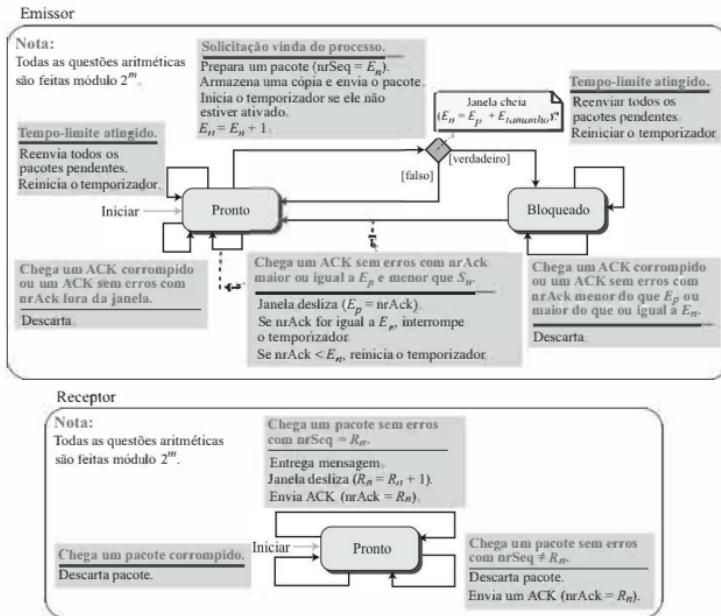



Figura 3.27 MEFs para o protocolo *Go-Back-N*.

Emissor

O emissor começa no estado *pronto*, mas depois disso ele pode se manter em um de dois estados: *pronto* ou *bloqueado*. As duas variáveis são normalmente inicializadas em 0 ($E_p = E_n = 0$).

- **Estado pronto.** Quatro eventos podem ocorrer quando o emissor está no estado *pronto*.

  a. Se um pedido vier da camada de aplicação, o emissor cria um pacote com o número de sequência igual a E_n. Uma cópia do pacote é armazenada e o pacote é então enviado. O emissor também inicializa o temporizador caso ele já não esteja ativado. O valor de E_n é então incrementado, ($E_n = E_n + 1$) módulo 2^m. Se a janela estiver cheia, $E_n = (E_p + E_{tamanho})$ módulo 2^m, o emissor vai para o estado *bloqueado*.

  b. Se for recebido umACK sem erros cujo valor de nrAck corresponde a um dos pacotes pendentes, o emissor desliza a janela (faz E_p = nrAck) e, se todos os pacotes pendentes forem confirmados (nrAck = E_n), o temporizador é interrompido. Se nem todos os pacotes pendentes forem confirmados, o temporizador é reiniciado.

  c. Se chegar um ACK corrompido, ou um ACK livre de erros cujo nrAck não corresponde ao pacote pendente, ele é descartado.

  d. Se o tempo-limite for atingido, o emissor reenvia todos os pacotes pendentes e reinicia o temporizador.

- **Estado bloqueado.** Três eventos podem ocorrer nesse caso:
  - **a.** Se for recebido um ACK livre de erros cujo nrAck corresponde a um dos pacotes pendentes, o emissor desliza a janela (faz E_p = nrAck) e, se todos os pacotes pendentes tiverem sido confirmados (nrAck = E_n), o temporizador é interrompido. Se nem todos os pacotes pendentes tiverem sido confirmados, o temporizador é reiniciado. O emissor, em seguida, vai para o estado *pronto*.
  - **b.** Se for recebido um ACK corrompido ou um ACK livre de erros cujo nrAck não corresponde a um dos pacotes pendentes, o ACK é descartado.
  - **c.** Se o tempo-limite for atingido, o emissor reenvia todos os pacotes pendentes e reinicia o temporizador.

Receptor

O receptor está sempre no estado *pronto*. A única variável R_n é inicializada com o valor 0. Três eventos podem ocorrer:

- **a.** Se for recebido um pacote livre de erros com nrSeq R_n, a mensagem no pacote é entregue à camada de aplicação. A janela então desliza, $R_n = (R_n + 1)$ módulo 2^m. Finalmente, um ACK é enviado com nrAck = R_n.
- **b.** Se for recebido um pacote livre de erros cujo valor de nrSeq encontra-se fora da janela, o pacote é descartado, mas um ACK com nrAck = R_n é enviado.
- **c.** Se for recebido um pacote corrompido, ele é descartado.

Tamanho da janela de envio

Podemos agora mostrar porque o tamanho da janela de envio deve ser inferior a 2^m. Como exemplo, escolhemos $m = 2$, o que significa que o tamanho da janela pode ser $2^m - 1$, ou seja, 3. A Figura 3.28 compara uma janela de tamanho 3 com outra de tamanho 4. Se o tamanho da janela for 3 (menor que 2^m) e as três confirmações forem todas perdidas, o tempo-limite do temporizador é atingido e todos os três pacotes são reenviados. O receptor fica, então, esperando o pacote 3, não o pacote 0, de

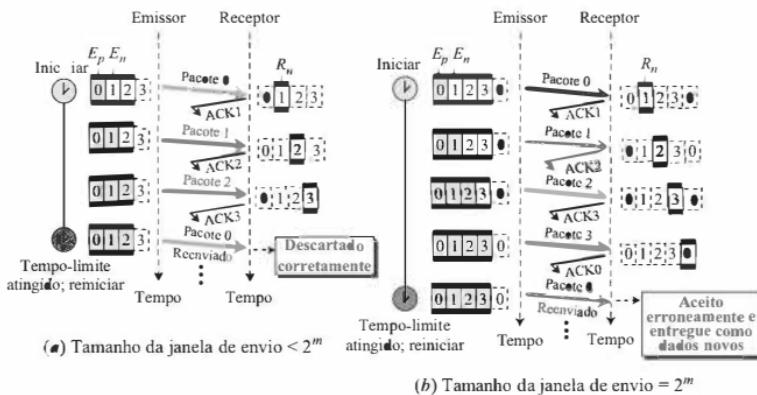



Figura 3.28 Tamanho da janela de envio para o protocolo *Go-Back-N*.

modo que o pacote duplicado é corretamente descartado. Por outro lado, se o tamanho da janela for igual a 4 (igual a 2^2) e todas as confirmações forem perdidas, o emissor enviará uma cópia do pacote 0. Entretanto, nesse momento, a janela do receptor espera receber o pacote 0 (no ciclo seguinte), de modo que ela aceita o pacote 0 não como uma cópia reenviada, mas como o primeiro pacote do ciclo seguinte. Isto é um erro que mostra que o tamanho da janela de envio sempre deve ser inferior a 2^m.

No protocolo *Go-Back-N*, o tamanho da janela de envio deve ser inferior a 2^m; o tamanho da janela de recepção é sempre 1.

Exemplo 3.7

A Figura 3.29 mostra um exemplo do protocolo *Go-Back-N*. Esse é um exemplo de um caso no qual o canal de envio é confiável, mas o canal de recepção não o é. Não há perda de pacotes de dados, mas alguns ACKs chegam atrasados e um deles é perdido. O exemplo também mostra como as confirmações cumulativas podem ajudar se as confirmações chegarem atrasadas ou forem perdidas.

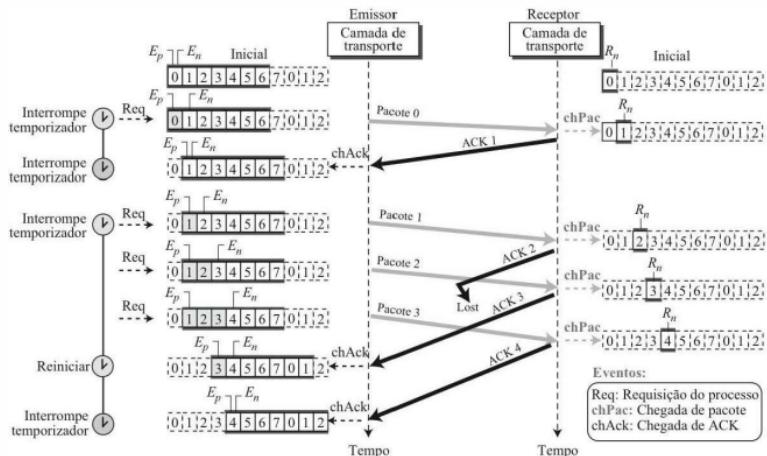



Figura 3.29 Diagrama de fluxo para o Exemplo 3.7.

Após a inicialização, ocorrem alguns eventos no emissor. Eventos de requisição são acionados por blocos de mensagens vindos da camada de aplicação; eventos de chegada são acionados por ACKs recebidos da camada de rede. Não há eventos de expiração do tempo-limite nesse exemplo porque todos os pacotes pendentes são confirmados antes do temporizador atingir o tempo-limite. Perceba que embora o ACK 2 seja perdido, o ACK 3 é cumulativo e serve tanto para o ACK 2 como para o ACK 3. Há quatro eventos no lado do receptor.

Exemplo 3.8

A Figura 3.30 mostra o que acontece quando um pacote é perdido. Os pacotes 0, 1, 2 e 3 são enviados. No entanto, o pacote 1 é perdido. O receptor recebe os pacotes 2 e 3, mas eles são descartados porque eles são

recebidos fora de ordem (o pacote 1 é o pacote esperado). Quando o receptor recebe os pacotes 2 e 3, ele envia ACK 1 para mostrar que está esperando receber o pacote 1. No entanto, esses ACKs não são úteis para o emissor porque o nrAck é igual a E_p e não é maior que E_p. Assim, o emissor irá descartá-los. Quando o tempo--limite é atingido, o emissor reenvia os pacotes 1, 2 e 3 e estes são, então, confirmados.

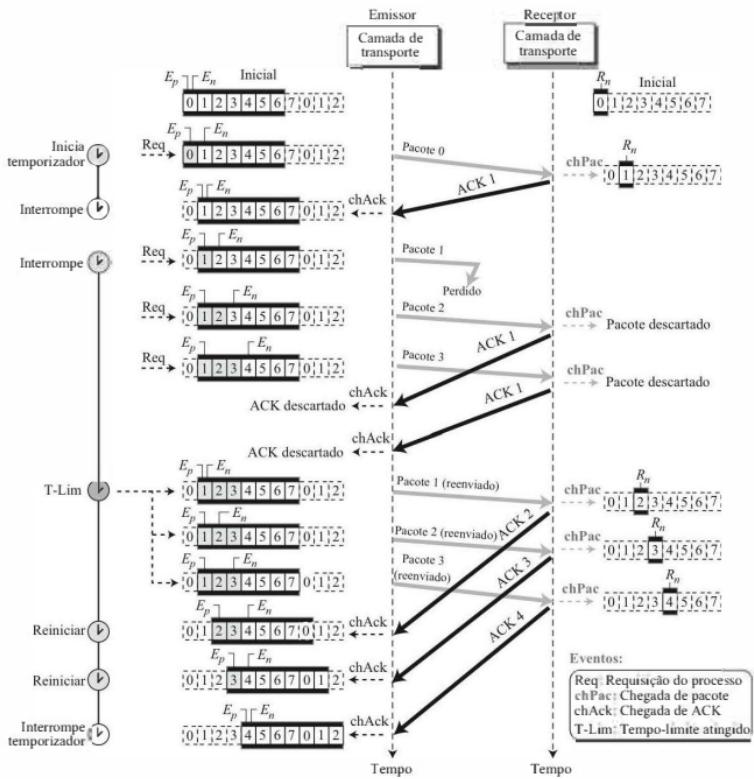



Figura 3.30 Diagrama de fluxo para o Exemplo 3.8.

Go-Back-N versus *Stop-and-Wait*

O leitor pode ter percebido que há uma semelhança entre os protocolos *Go-Back-N* e *Stop-and--Wait*. O protocolo *Stop-and-Wait*, na verdade, é um protocolo *Go-Back-N* no qual existem apenas dois números de sequência e o tamanho da janela de envio é igual a 1. Em outras palavras, $m = 1$ e $2^m - 1 = 1$. No protocolo *Go-Back-N*, dizemos que a aritmética é modulo 2^m; no *Stop-and-Wait* ela é módulo 2, valor equivalente a 2^m quando $m = 1$.

3.2.4 Protocolo de repetição seletiva

O protocolo *Go-Back-N* simplifica o processo no lado do receptor. O receptor mantém o registro de apenas uma variável e não é necessário colocar pacotes fora de ordem em um *buffer*; esses pacotes são simplesmente descartados. No entanto, esse protocolo é ineficiente se o protocolo de rede subjacente perder diversos pacotes. Cada vez que um único pacote for perdido ou corrompido, o emissor reenvia todos os pacotes pendentes, embora alguns deles possam ter sido recebidos sãos e salvos, mas fora de ordem. Se a camada de rede estiver perdendo muitos pacotes por causa de congestionamento na rede, o reenvio de todos esses pacotes pendentes piorará o congestionamento e, eventualmente, mais pacotes serão perdidos. Isso tem um efeito avalanche que pode resultar no colapso total da rede.

Outro protocolo, denominado **protocolo Repetição Seletiva** (RS), foi concebido para, como o nome indica, reenviar apenas pacotes selecionados, que são aqueles que foram realmente perdidos. A estrutura geral desse protocolo é mostrada na Figura 3.31.

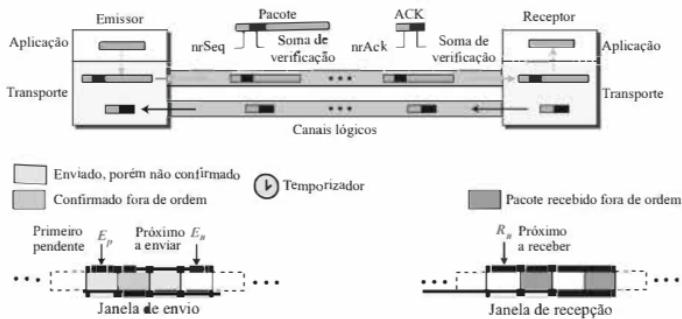



Figura 3.31 Visão geral da Repetição Seletiva.

Janelas

O protocolo Repetição Seletiva também utiliza duas janelas: uma janela de envio e uma de recepção. No entanto, existem diferenças entre as janelas desse protocolo e as janelas do *Go-Back-N*. Em primeiro lugar, o tamanho máximo da janela de envio é muito menor; ele vale 2^{m-1}. A razão para isso será discutida mais adiante. Em segundo lugar, a janela de recepção tem o mesmo tamanho que a janela de envio.

O tamanho máximo da janela de envio pode ser 2^{m-1}. Por exemplo, se $m = 4$, os números de sequência podem ir de 0 a 15, mas o tamanho máximo da janela é de apenas 8 (ele vale 15 no protocolo *Go-Back-N*). Mostramos a janela de envio do protocolo de Repetição Seletiva na Figura 3.32 para enfatizar o seu tamanho.

A janela de recepção no protocolo de Repetição Seletiva é totalmente diferente daquela usada no protocolo *Go-BackN*. O tamanho da janela de recepção é igual ao tamanho da janela de envio (no máximo 2^{m-1}). O protocolo de Repetição Seletiva permite que uma quantidade de pacotes equivalente ao tamanho da janela de recepção chegue fora de ordem e que eles sejam armazenados até que haja um conjunto de pacotes consecutivos que possam ser entregues à camada de aplicação. Como o tamanho da janela de envio e da janela de recepção é o mesmo, todos os pacotes enviados podem chegar fora de ordem e serem armazenados até que possam ser entregues. Precisamos, no entanto, ressaltar que em um protocolo confiável o receptor *nunca* entrega pacotes fora de ordem

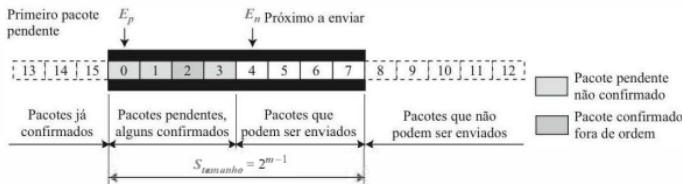



Figura 3.32 Janela de envio para o protocolo de Repetição Seletiva.

para a camada de aplicação. A Figura 3.33 mostra a janela de recepção no protocolo de Repetição Seletiva. As posições dentro da janela que estão sombreadas indicam os pacotes que chegaram fora de ordem e que estão esperando a chegada de pacotes transmitidos anteriormente para que possam ser entregues à camada de aplicação.

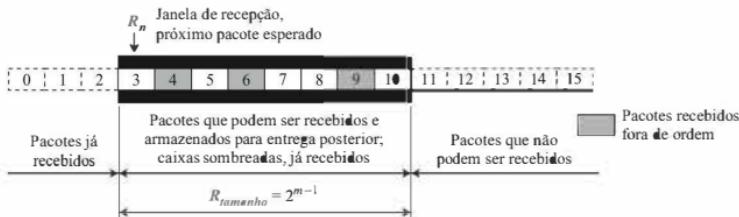



Figura 3.33 Janela de recepção para o protocolo de Repetição Seletiva.

Temporizador

Teoricamente, o protocolo de Repetição Seletiva usa um temporizador para cada pacote pendente. Quando o tempo-limite é atingido, somente o pacote correspondente é reenviado. Em outras palavras, o GBN trata os pacotes pendentes como um grupo; o protocolo RS os trata individualmente. No entanto, a maioria dos protocolos da camada de transporte que implementa o protocolo RS usa um único temporizador. Por isso, aqui usamos apenas um temporizador.

Confirmações

Existe ainda outra diferença entre os dois protocolos. No GBN, um nrAck é cumulativo; ele indica o número de sequência do próximo pacote esperado, confirmando que todos os pacotes anteriores foram recebidos sãos e salvos. A semântica de confirmação é diferente no protocolo RS. No protocolo RS, um nrAck define o número de sequência de um único pacote que foi recebido são e salvo; ele não carrega informações relativas a qualquer outro pacote.

No protocolo de Repetição Seletiva, um número de confirmação define o número de sequência do pacote recebido livre de erros.

Exemplo 3.9

Considere que um emissor envia seis pacotes: 0, 1, 2, 3, 4 e 5. O emissor recebe um ACK com nrAck = 3. Qual é a interpretação se o sistema estiver usando GBN? E se ele estiver usando RS?

Solução

Se o sistema utilizar GBN, isso significa que os pacotes 0, 1 e 2 foram recebidos sem corrupção (livre de erros) e o receptor está esperando o pacote 3. Se o sistema estiver usando RS, significa que o pacote 3 foi recebido livre de erros; o ACK não diz nada sobre os outros pacotes.

MEFs

A Figura 3.34 apresenta as MEFs para o protocolo de Repetição Seletiva. Elas são similares àquelas do GBN, mas há algumas diferenças.

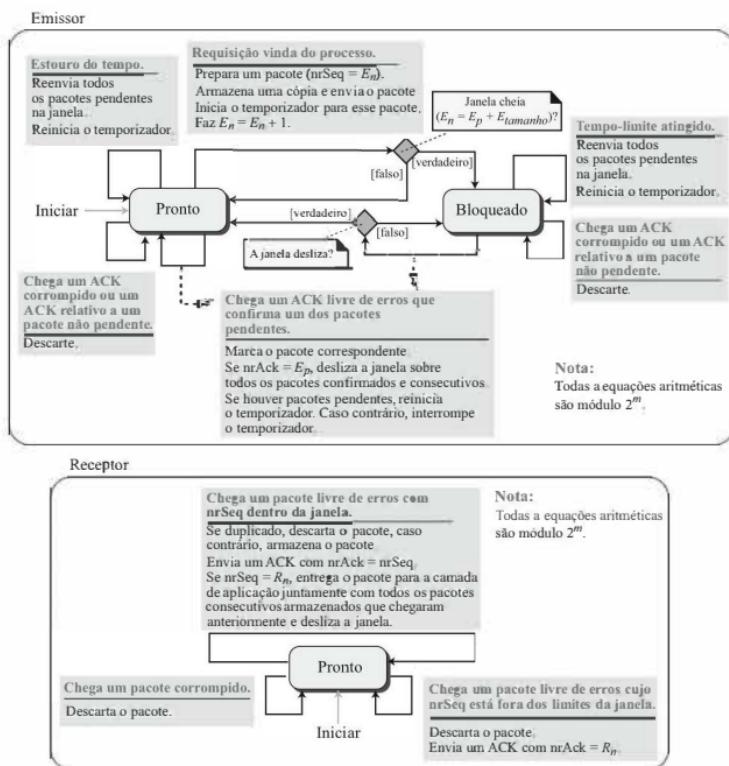



Figura 3.34 MEFs para o protocolo de Repetição Seletiva (RS).

Emissor

O emissor começa no estado *pronto*, porém mais tarde ele pode estar em um de dois estados: *pronto* ou *bloqueado*. A seguir, são mostrados os eventos e as ações correspondentes de cada estado.

- **Estado pronto.** Quatro eventos podem ocorrer nesse caso:
  - **a.** Se um pedido vier da camada de aplicação, o emissor cria um pacote cujo número de sequência vale E_n. Uma cópia do pacote é armazenada e o pacote é enviado. Se o temporizador não estiver sendo executado, o emissor inicia o temporizador. O valor de E_n é então incrementado, fazendo-se $E_n = (E_n + 1)$ módulo 2^m. Se a janela estiver cheia, ou seja, se $E_n = (E_p + E_{tamanho})$ módulo 2^m, o emissor passa para o estado *bloqueado*.
  - **b.** Se for recebido um ACK livre de erros cujo nrAck corresponde a um dos pacotes pendentes, esse pacote é marcado como confirmado. Se tivermos nrAck = E_p, a janela desliza para a direita até que E_p passe a apontar para o primeiro pacote não confirmado (todos os pacotes consecutivos confirmados estão agora fora da janela). Se houver pacotes pendentes, o temporizador é reiniciado; caso contrário, é interrompido.
  - **c.** Se for recebido um ACK corrompido ou um ACK livre de erros cujo nrAck não corresponda a pacotes pendentes, ele é descartado.
  - **d.** Se o tempo-limite for atingido, o emissor envia todos os pacotes não confirmados da janela e reinicia o temporizador.
- **Estado bloqueado.** Três eventos podem ocorrer neste caso:
  - **a.** Se for recebido um ACK livre de erros cujo valor de nrAck corresponda a um dos pacotes pendentes, esse pacote é marcado como confirmado. Além disso, se nrAck = E_p, a janela é deslizada para a direita até que E_p passe a apontar para o primeiro pacote não confirmado (todos os pacotes consecutivos confirmados estão agora fora da janela). Se a janela já deslizou, o emissor passa para o estado *pronto*.
  - **b.** Se for recebido um ACK corrompido ou um ACK livre de erros cujo nrAck não corresponda a pacotes pendentes, o ACK é descartado.
  - **c.** Se o tempo-limite for atingido, o emissor envia todos os pacotes não confirmados da janela e reinicia o temporizador.

Receptor

O receptor está sempre no estado *pronto*. Três eventos podem ocorrer:

- **a.** Se for recebido um pacote livre de erros cujo valor de nrSeq esteja dentro da janela, o pacote é armazenado e um ACK com nrAck = nrSeq é enviado. Além disso, se nrSeq = R_n, então o pacote e todos os pacotes consecutivos que chegaram anteriormente são entregues à camada de aplicação e a janela desliza de forma que R_n passe a apontar para a primeira posição vazia da janela.
- **b.** Se for recebido um pacote livre de erros cujo valor de nrSeq esteja fora da janela, o pacote é descartado e um ACK com nrAck = R_n é retornado para o emissor. Isto é necessário para permitir que o emissor faça a sua janela deslizar caso alguns ACKs relativos a pacotes com nrSeq < R_n tenham sido perdidos.
- **c.** Se for recebido um pacote corrompido, ele é descartado.

Exemplo 3.10

Esse exemplo é semelhante ao Exemplo 3.8 (Figura 3.30), no qual o pacote 1 é perdido. Mostramos como o protocolo de Repetição Seletiva se comporta nesse caso. A Figura 3.35 ilustra a situação.

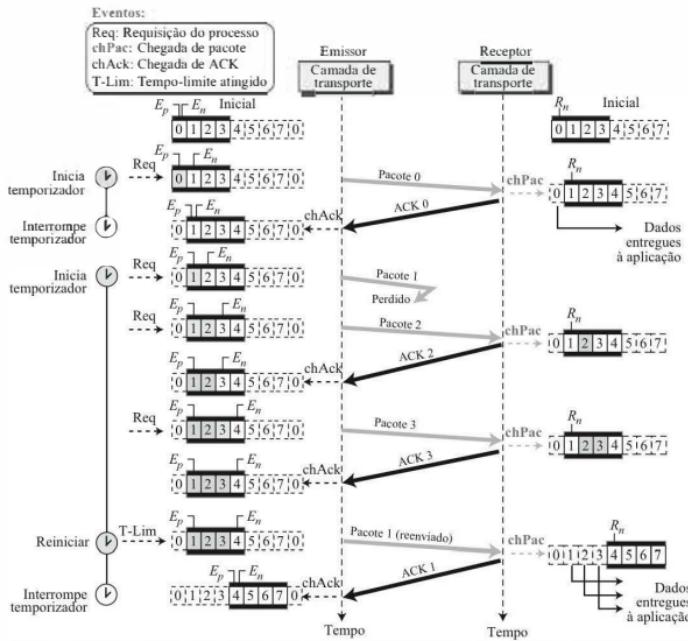



Figura 3.35 Diagrama de fluxo para o Exemplo 3.10.

No emissor, o pacote 0 é transmitido e confirmado. O pacote 1 é perdido. Os pacotes 2 e 3 chegam fora de ordem e são confirmados. Quando o tempo-limite do temporizador é atingido, o pacote 1 (o único pacote não confirmado) é reenviado e confirmado. Em seguida, a janela de envio desliza.

No lado do receptor, precisamos distinguir entre a aceitação de um pacote e a sua entrega à camada de aplicação. Na segunda chegada, o pacote 2 é recebido e então armazenado e marcado (posição sombreada na janela), mas não pode ser entregue porque o pacote 1 não foi recebido. Na chegada seguinte, o pacote 3 é recebido e também marcado e armazenado, mas nenhum dos pacotes pode ser entregue ainda. Apenas na última chegada, quando finalmente é recebida uma cópia do pacote 1, os pacotes 1, 2 e 3 podem ser entregues à camada de aplicação. Há duas condições para isso: a primeira é que um conjunto de pacotes consecutivos deve ter chegado, e a segunda é que o conjunto de pacotes comece a partir do início da janela. Após a chegada do primeiro pacote, há apenas um pacote e ele encontra-se no início da janela. Após a última chegada, existem três pacotes e o primeiro deles encontra-se no início da janela. O ponto-chave é que uma camada de transporte confiável se compromete a entregar os pacotes em ordem.

Tamanhos de janela

Podemos agora mostrar por que o tamanho das janelas do emissor e do receptor pode ser no máximo metade de 2^m. Por exemplo, escolhemos $m = 2$, o que significa que o tamanho da janela é $2^m/2$ ou $2^{(m-1)} = 2$. A Figura 3.36 compara uma janela de tamanho 2 com uma de tamanho 3.

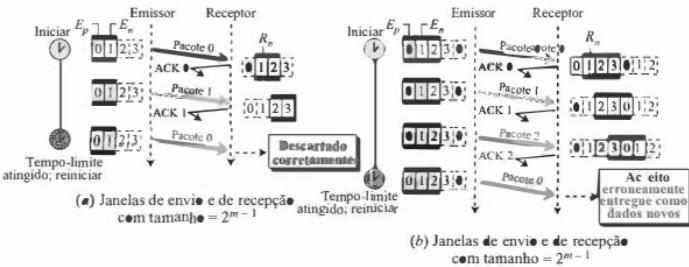



Figura 3.36 Repetição Seletiva: tamanho da janela.

Se o tamanho da janela for 2 e todas as confirmações forem perdidas, o temporizador para o pacote 0 expira e o pacote 0 é reenviado. No entanto, a janela do receptor está agora à espera do pacote 2, não do pacote 0, de modo que esse pacote duplicado será corretamente descartado (o número de sequência 0 não está na janela). Quando o tamanho da janela é 3 e todas as confirmações são perdidas, o emissor envia uma cópia do pacote 0. No entanto, dessa vez, a janela do receptor espera receber o pacote 0 (0 faz parte da janela), de modo que o pacote 0 é aceito não como um pacote duplicado, mas como um pacote do ciclo seguinte. Isto é claramente um erro.

Na Repetição Seletiva, o tamanho das janelas de envio e de recepção pode ser, no máximo, metade de 2^m.

3.2.5 Protocolos bidirecionais: mecanismo de carona

Os quatro protocolos que discutimos anteriormente nesta seção são unidirecionais: o fluxo de dados dos pacotes se dá em uma só direção e as confirmações trafegam na outra. Na vida real, os pacotes de dados normalmente fluem em ambos os sentidos: do cliente para o servidor e do servidor para o cliente. Isto significa que as confirmações também precisam fluir em ambas as direções. Uma técnica chamada ***piggybacking***, ou **mecanismo de carona**, é usada para melhorar a eficiência dos protocolos bidirecionais. Quando um pacote estiver transportando dados de A para B, ele também pode carregar informações de confirmação para pacotes que chegaram vindos de B; quando um pacote estiver transportando dados de B para A, ele também pode carregar confirmações para os pacotes recebidos de A.

A Figura 3.37 mostra uma visão geral do protocolo GBN implementado bidirecionalmente usando *piggybacking*. O cliente e o servidor usam, cada um, duas janelas independentes: de envio e de recepção.

3.2.6 Protocolos da camada de transporte da Internet

Agora que discutimos o princípio geral por trás da camada de transporte, nas próximas duas seções nos concentraremos nos protocolos de transporte da Internet. Embora a Internet use vários protocolos da camada de transporte, discutimos apenas dois deles neste capítulo, conforme mostra a Figura 3.38.

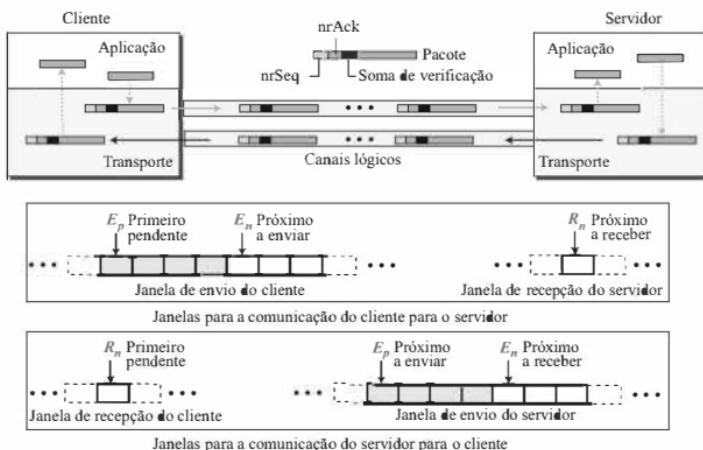



Figura 3.37 Modelo do mecanismo de carona (*piggybacking*) no protocolo *Go-Back-N*.

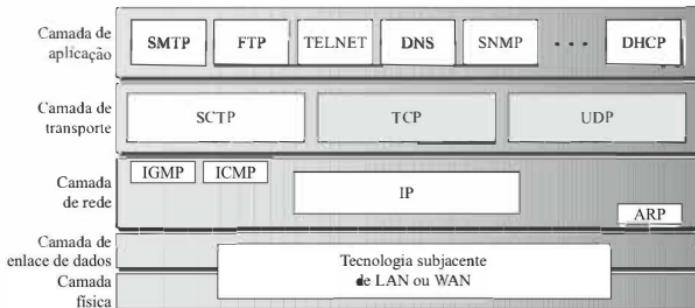
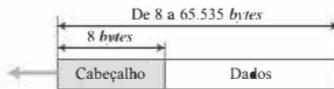



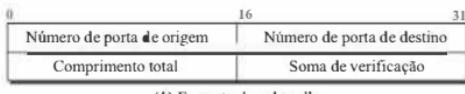
Figura 3.38 Posição da camada de transporte na pilha de protocolos TCP/IP.

A figura mostra o relacionamento de dois protocolos da camada de transporte, UDP e TCP, com os outros protocolos e camadas da pilha de protocolos TCP/IP. Esses protocolos estão localizados entre a camada de aplicação e a camada de rede e servem como intermediários entre os programas de aplicação e as operações de rede.

O UDP é um protocolo não orientado à conexão e não confiável da camada de transporte, sendo utilizado por sua simplicidade e eficiência em aplicações nas quais o controle de erros pode ser fornecido pelo processo na camada de aplicação. O TCP é um protocolo orientado à conexão e confiável que pode ser utilizado em qualquer aplicação na qual a confiabilidade seja importante. Existem outros protocolos da camada de transporte, como o SCTP, que discutimos em outros capítulos.

Conforme discutido anteriormente, um protocolo da camada de transporte geralmente tem várias responsabilidades. Uma delas é criar uma comunicação processo a processo; esses protocolos utilizam números de porta para fazer isso (Tabela 3.1).

Tabela 3.1 Algumas portas bem conhecidas usadas com o UDP e com o TCP.

Porta	Protocolo	UDP	TCP	Descrição
7	*Echo*	√		Ecoa de volta um datagrama recebido
9	*Discard*	√		Descarta qualquer datagrama recebido
11	*Users*	√	√	Usuários ativos
13	*Daytime*	√	√	Retorna data e hora
17	*Quote*	√	√	Retorna um texto em ASCII
19	*Chargen*	√	√	Retorna uma sequência de caracteres
20, 21	FTP		√	Protocolo de Transferência de Arquivos (File Transfer Protocol)
23	TELNET		√	Terminal de Rede (Terminal Network)
25	SMTP		√	Protocolo Simples de Transferência de Correio (Simple Mail Transfer Protocol)
53	DNS	√	√	Sistema de Nomes de Domínio (Domain Name System)
67	DHCP	√	√	Protocolo de Configuração Dinâmica de *Host* (Dynamic Host Configuration Protocol)
69	TFTP	√		Protocolo Trivial de Transferência de Arquivos (Trivial File Transfer Protocol)
80	HTTP		√	Protocolo de Transferência de Hipertexto (Hypertext Transfer Protocol)
111	RPC	√	√	Chamada de Procedimento Remoto (Remote Procedure Call)
123	NTP	√	√	Protocolo de Tempo para Redes (Network Time Protocol)
161, 162	SNMP		√	Protocolo Simples de Gerenciamento de Rede (Simple Network Management Protocol)

3.3 PROTOCOLO DE DATAGRAMAS DE USUÁRIO

O **Protocolo de Datagramas de Usuário** (UDP – User Datagram Protocol) é um protocolo de transporte não confiável e não orientado à conexão. Ele não acrescenta nada aos serviços do IP, exceto pelo fornecimento de comunicação processo a processo em vez de comunicação *host* a *host*. Se o UDP é tão pouco poderoso, porque um processo o usaria? Com as desvantagens vêm algumas vantagens. O UDP é um protocolo muito simples, que tem um uso mínimo de recursos. Se um processo quiser enviar uma mensagem pequena e não se importar muito com a confiabilidade da transmissão, ele pode usar o UDP. Enviar uma mensagem pequena usando o UDP exige muito menos interação entre o emissor e o receptor do que se o TCP for usado. Discutimos algumas aplicações do UDP no fim desta seção.

3.3.1 Datagramas de usuário

Os pacotes UDP, denominados **datagramas de usuário** (*user datagrams*), têm um cabeçalho com o tamanho fixo de 8 *bytes* constituído por quatro campos, cada um de 2 *bytes* (16 *bits*). A Figura 3.39 mostra o formato de um datagrama de usuário. Os dois primeiros campos definem os números de porta de origem e de destino. O terceiro campo define o comprimento total do datagrama, considerando cabeçalho mais dados. Os 16 *bits* podem definir um comprimento total de 0 a 65.535 *bytes*. No entanto, o comprimento total deve ser menor, pois um datagrama de usuário UDP é armazenado em um datagrama IP, que possui um comprimento máximo de 65.535 *bytes*. O último campo pode carregar uma soma de verificação (*checksum*) que é opcional, conforme explicado mais adiante.

Figura 3.39 Formato do pacote do datagrama de usuário.

Exemplo 3.11

A seguir, mostramos o conteúdo de um cabeçalho UDP no formato hexadecimal.

CB84000D001C001C

a. Qual é o número da porta de origem?
b. Qual é o número da porta de destino?
c. Qual é o comprimento total do datagrama de usuário?
d. Qual é o comprimento dos dados?
e. O pacote está indo do cliente para o servidor ou no sentido contrário?
f. Qual é o processo-cliente?

Solução

a. O número da porta de origem corresponde aos quatro primeiros dígitos hexadecimais $(CB84)_{16}$, o que significa que o número da porta de origem é 52100.
b. O número da porta de destino corresponde ao segundo conjunto de quatro dígitos hexadecimais $(000D)_{16}$, o que significa que o número da porta de destino é 13.
c. O terceiro conjunto de quatro dígitos hexadecimais $(001C)_{16}$ informa o comprimento do pacote UDP inteiro, que nesse caso é de 28 *bytes*.
d. O comprimento dos dados corresponde ao comprimento do pacote completo menos o do cabeçalho, ou seja, 28 – 8 = 20 *bytes*.
e. Como o número da porta de destino é 13 (porta bem conhecida), o pacote está indo do cliente para o servidor.
f. O processo-cliente é o *Daytime* (ver Tabela 3.1).

3.3.2 Serviços do UDP

Anteriormente, discutimos os serviços gerais prestados por um protocolo da camada de transporte. Nesta seção, discutimos quais partes desses serviços gerais são fornecidas pelo UDP.

Comunicação processo a processo

O UDP fornece comunicação processo a processo usando endereços de *sockets*, uma combinação de endereços IP e números de porta.

Serviços não orientados à conexão

Conforme mencionado anteriormente, o UDP oferece um *serviço não orientado à conexão*. Isto significa que cada datagrama enviado pelo usuário é um datagrama UDP independente. Não há uma relação entre os diferentes datagramas de usuário, mesmo se eles forem provenientes do mesmo processo de origem e estiverem indo para o mesmo programa de destino. Os datagramas de usuário não são numerados. Além disso, ao contrário do TCP, não há estabelecimento de conexão e não há finalização da conexão. Isto significa que cada datagrama de usuário pode viajar por um caminho diferente.

Uma das consequências de ser não orientado à conexão é que o processo que usa o UDP não pode enviar um fluxo de dados e esperar que o quebre em diferentes datagramas de usuário. Na verdade, cada pedido deve ser suficientemente pequeno para caber em um datagrama de usuário. Apenas os processos que enviam mensagens curtas, mensagens menores do que 65.507 *bytes* (65.535 menos os 8 *bytes* do cabeçalho UDP e menos 20 *bytes* do cabeçalho IP), podem usar o UDP.

Controle de fluxo

O UDP é um protocolo muito simples. Não há *controle de fluxo* e, portanto, nenhum mecanismo de janelas. O receptor pode ficar sobrecarregado com as mensagens recebidas. A ausência de controle de fluxo significa que o processo que está usando o UDP deve fornecer esse serviço, se necessário.

Controle de erros

Não existe um mecanismo de *controle de erros* no UDP além da soma de verificação (*checksum*). Isto significa que o emissor não sabe se uma mensagem foi perdida ou duplicada. Quando o receptor detecta um erro por meio da soma de verificação, o datagrama de usuário é descartado silenciosamente. A ausência de um mecanismo de controle de erros significa que o processo que está usando o UDP deve fornecer esse serviço, se necessário.

Soma de verificação

Discutimos a soma de verificação (*checksum*) e seu cálculo no Capítulo 5. O cálculo da soma de verificação no UDP inclui três seções: um pseudocabeçalho, o cabeçalho UDP e os dados que chegam da camada de aplicação. O *pseudocabeçalho* é a parte do cabeçalho do pacote IP (discutido no Capítulo 4) no qual o datagrama de usuário é encapsulado, com alguns campos fixados em 0s (ver Figura 3.40).

Se a soma de verificação não incluísse o pseudocabeçalho, um datagrama de usuário poderia chegar são e salvo. No entanto, se o cabeçalho IP fosse corrompido, o pacote poderia ser entregue ao destino errado.

O campo de *protocolo* é adicionado para assegurar que o pacote pertence ao UDP, e não ao TCP. Veremos mais tarde que se um processo puder usar tanto o UDP como TCP, o número de porta de destino pode ser o mesmo. O valor do campo de protocolo para o UDP é 17. Se esse valor for alterado durante a transmissão, o cálculo da soma de verificação no receptor detectará o erro e o UDP descartará o pacote. Este não é entregue ao protocolo errado.

Pseudocabeçalho	Endereço IP de origem (32 *bits*)		
	Endereço IP de destino (32 *bits*)		
	Repleto de 0s	Protocolo (8 *bits*)	Comprimento total UDP (16 *bits*)
Cabeçalho	Endereço da porta de origem 16 *bits*		Endereço da porta de destino 16 *bits*
	Comprimento total UDP 16 *bits*		Soma de verificação 16 *bits*
	Dados (Devem ser adicionados *bits* de preenchimento/*padding* para fazer com que o tamanho dos dados seja múltiplo de 16 *bits*)		

Figura 3.40 Pseudocabeçalho para o cálculo da soma de verificação (*checksum*).

Inclusão opcional da soma de verificação

O emissor de um pacote UDP pode optar por não calcular a soma de verificação. Nesse caso, o campo de soma de verificação é preenchido com 0s antes de ser enviado. Na situação em que o emissor decide calcular a soma de verificação, porém o resultado do cálculo acaba sendo composto apenas por 0s, a soma de verificação é alterada para uma sequência repleta de 1s antes do envio do pacote. Ou seja, o emissor complementa a soma duas vezes. Perceba que essa alteração não cria confusão, porque o valor da soma de verificação nunca é formado apenas por 1s em uma situação normal (ver exemplo a seguir).

Exemplo 3.12

Qual é o valor enviado no campo de soma de verificação em cada uma das seguintes situações hipotéticas?

a. O emissor decide não incluir a soma de verificação.

b. O emissor decide incluir a soma de verificação, mas o valor da soma é uma sequência repleta de 1s.

c. O emissor decide incluir a soma de verificação, mas o valor da soma é uma sequência repleta de 0s.

Solução

a. O valor atribuído à soma de verificação é uma sequência repleta de 0s para mostrar que a soma de verificação não foi calculada.

b. Quando o emissor complementa a soma, o resultado é uma sequência repleta de 0s; o emissor complementa o resultado novamente antes de enviar os dados. O valor enviado no campo de soma de verificação é uma sequência repleta de 1s. A segunda operação de complemento é necessária para evitar confusão com o caso descrito no item *a*.

c. Essa situação nunca acontece porque implica que todos os valores de cada termo do cálculo da soma sejam sequências de 0s, o que é impossível, visto que alguns campos do pseudocabeçalho têm valores diferentes de zero.

Controle de congestionamento

Como o UDP é um protocolo não orientado à conexão, ele não fornece controle de congestionamento. O UDP considera que os pacotes enviados são pequenos e esporádicos, não sendo capazes de criar congestionamento na rede. Essa suposição pode ou não ser verdadeira hoje, já que o UDP é atualmente usado para a transferência em tempo real de áudio e vídeo interativos.

Encapsulamento e desencapsulamento

Para enviar uma mensagem de um processo para outro, o protocolo UDP encapsula e desencapsula mensagens.

Enfileiramento

Falamos sobre o conceito de portas sem discutir uma implementação de fato das mesmas. No UDP, as filas são associadas com as portas.

No lado do cliente, quando um processo é iniciado, ele solicita um número de porta para o sistema operacional. Algumas implementações criam tanto uma fila de entrada como uma de saída associadas a cada processo. Outras implementações criam apenas uma fila de entrada associada a cada processo.

Multiplexação e demultiplexação

Em uma estação executando a pilha de protocolos TCP/IP, há apenas uma camada UDP, mas possivelmente vários processos que podem querer utilizar os serviços do UDP. Para lidar com essa situação, o UDP realiza multiplexação e demultiplexação.

Comparação entre o UDP e um protocolo genérico simples

Podemos comparar o UDP com o protocolo não orientado à conexão simples que discutimos anteriormente. A única diferença é que o UDP fornece uma soma de verificação opcional para detectar corrupção nos pacotes no lado do receptor. Se a soma de verificação for adicionada ao pacote, a camada UDP no receptor pode verificar o pacote e descartá-lo se ele estiver corrompido. No entanto, nenhuma informação a respeito é enviada para o emissor.

O UDP é um exemplo de protocolo simples não orientado à conexão que discutimos anteriormente, exceto por uma soma de verificação opcional que pode ser adicionado aos pacotes para permitir a detecção de erros.

3.3.3 Aplicações do UDP

Embora o UDP não se enquadre em quase nenhum dos critérios mencionados anteriormente para um protocolo de camada de transporte confiável, o UDP é preferível em algumas aplicações. A razão é que alguns serviços podem ter efeitos secundários inaceitáveis ou não preferíveis. Um projetista de aplicação às vezes precisa fazer concessões para obter algo melhor. Por exemplo, no dia a dia, todos sabem que uma transportadora que demora um dia para entregar um pacote cobra um valor mais alto do que um serviço de entrega que demora três dias. Embora baixo tempo e baixo custo sejam duas características desejáveis em uma entrega, eles são requisitos conflitantes entre si. Precisamos escolher o que melhor se encaixa às nossas necessidades.

Nesta seção, primeiro discutimos algumas características do UDP que precisam ser consideradas quando se projeta um programa de aplicação e, em seguida, mostramos algumas aplicações típicas.

Características do UDP

Discutimos aqui brevemente algumas características do UDP e as suas vantagens e desvantagens.

Serviço não orientado à conexão

Conforme mencionamos anteriormente, o UDP é um protocolo não orientado à conexão. Cada pacote UDP é independente dos outros pacotes enviados pelo mesmo programa de aplicação. Essa característica pode ser considerada uma vantagem ou desvantagem, dependendo dos requisitos da aplicação. É uma vantagem se, por exemplo, uma aplicação-cliente precisar enviar um pedido curto para um servidor e receber uma resposta curta. Se tanto a solicitação como a resposta couberem em um único datagrama de usuário, um serviço não orientado à conexão pode ser preferível. A carga de controle para estabelecer e finalizar uma conexão pode ser significativa nesse caso. Em um serviço orientado à conexão, para atingir o objetivo anterior, pelo menos nove pacotes são trocados entre o cliente e o servidor; já em um serviço não orientado à conexão, somente dois pacotes são trocados. O serviço não orientado à conexão fornece menos atraso; o orientado à conexão cria mais atraso. Se o atraso for uma questão importante para a aplicação, o serviço não orientado à conexão é preferível.

Exemplo 3.13

Uma aplicação cliente-servidor como o DNS (ver Capítulo 2) usa os serviços do UDP porque o cliente precisa enviar um pedido curto para um servidor e receber uma resposta rápida dele. Tanto o pedido como a resposta cabem em um datagrama de usuário. Como apenas uma mensagem é trocada em cada sentido, a característica de não haver conexão não é um problema; o cliente e o servidor não se importam que as mensagens sejam entregues fora de ordem.

Exemplo 3.14

Uma aplicação cliente-servidor como o SMTP (ver Capítulo 2), é utilizada no correio eletrônico, não podendo usar os serviços do UDP porque um usuário pode enviar uma mensagem de e-mail longa, que pode incluir conteúdo multimídia (imagens, áudio ou vídeo). Se o aplicativo usar o UDP e a mensagem não couber em um único datagrama de usuário, ela deve ser dividida pelo aplicativo em diferentes datagramas de usuários. Aqui, um serviço não orientado à conexão pode criar problemas. Os datagramas de usuário podem chegar e ser entregues fora de ordem à aplicação no receptor, que pode não ser capaz de reordenar os pedaços. Isto significa que o serviço não orientado à conexão apresenta uma desvantagem quando os aplicativos enviam mensagens longas. No SMTP, quando uma mensagem é enviada, não se espera receber uma resposta rapidamente (às vezes nenhuma resposta é necessária). Isto significa que o atraso extra inerente ao serviço orientado à conexão não é um problema crucial para o SMTP.

Ausência de controle de erros

O UDP não fornece controle de erros, mas um serviço não confiável. A maioria dos aplicativos espera receber um serviço confiável de um protocolo de camada de transporte. Embora um serviço confiável seja desejável, isso pode ter alguns efeitos secundários não aceitáveis para algumas aplicações. Quando uma camada de transporte fornece serviços confiáveis, se uma parte da mensagem for perdida ou corrompida, ela precisa ser reenviada. Isto significa que a camada de transporte no receptor pode não entregar essa parte à aplicação imediatamente; há um atraso desigual entre as diferentes partes da mensagem transmitidas à camada de aplicação. Algumas aplicações, por natureza, nem percebem esses atrasos irregulares, mas para outras, eles são cruciais.

Exemplo 3.15

Considere que estamos recebendo um arquivo de texto muito grande da Internet. Precisamos definitivamente usar uma camada de transporte que forneça um serviço confiável. Não queremos que alguma parte do arquivo esteja faltando ou tenha sido corrompida quando abrirmos o arquivo. O atraso criado entre as entregas das partes não é uma preocupação primordial para nós; podemos esperar até que todo o arquivo seja obtido antes de acessá-lo. Nesse caso, o UDP não é uma camada de transporte adequada.

Exemplo 3.16

Considere que estamos usando um aplicativo interativo em tempo real, como o Skype. Áudio e vídeo são divididos em quadros e enviados um após o outro. Se a camada de transporte for projetada para reenviar um quadro corrompido ou perdido, a sincronização de toda a transmissão pode ser perdida. O usuário de repente vê uma tela em branco e precisa esperar até que o quadro retransmitido chegue, o que não é tolerável. No entanto, se cada pequena parte da tela é enviada por meio de um único datagrama de usuário, a camada UDP no receptor pode facilmente ignorar o pacote corrompido ou perdido e entregar o restante dos pacotes para o programa da camada de aplicação. Essa parte da tela ficará em branco por um período muito curto, que a maioria dos usuários nem percebe.

Ausência de controle de congestionamento

O UDP não fornece controle de congestionamento. Entretanto, o UDP não cria tráfego adicional em uma rede propensa a erros. O TCP pode reenviar um pacote várias vezes e, assim, contribuir para a criação de congestionamentos ou piorá-los. Portanto, em alguns casos, a falta de controle de erros no UDP pode ser considerada uma vantagem quando o congestionamento é um grande problema.

Aplicações típicas

A seguir, são apresentadas algumas aplicações típicas que podem se beneficiar mais dos serviços do UDP do que daqueles oferecidos pelo TCP.

- O UDP é adequado para processos que necessitem de uma comunicação do tipo pedido e resposta simples, com pouca preocupação relativa a controle de fluxo e de erros. Ele não é normalmente utilizado para processos como o FTP, que precisa enviar dados em massa (ver Capítulo 2).
- O UDP é adequado para processos que possuam mecanismos internos para o controle de fluxo e de erros. Por exemplo, o Protocolo Trivial de Transferência de Arquivos (TFTP – Trivial File Transfer Protocol) inclui mecanismos próprios de controle de fluxo e de erros. Pode facilmente usar o UDP.
- O UDP é um protocolo de transporte adequado para comunicações via *multicast*. A capacidade de fazer *multicast* está incorporada no *software* do UDP, mas não no *software* do TCP.
- O UDP é utilizado para processos de gerenciamento como o SNMP (ver Capítulo 9).
- O UDP é usado para realizar algumas atualizações de rota em protocolos, como o Protocolo de Informações de Roteamento (RIP – Routing Information Protocol) (ver Capítulo 4).
- O UDP é normalmente utilizado para aplicações interativas em tempo real que não podem tolerar atrasos desiguais entre as seções de uma mensagem recebida (ver Capítulo 8).

3.4 PROTOCOLO DE CONTROLE DE TRANSMISSÃO

O **Protocolo de Controle de Transmissão** (TCP – Transmission Control Protocol) é um protocolo orientado à conexão e confiável. O TCP define explicitamente o estabelecimento da conexão, a transferência de dados e a fechamento para oferecer um serviço orientado à conexão. O TCP usa uma combinação de protocolos do tipo GBN e RS para fornecer confiabilidade. Para atingir esse objetivo, faz uso de mecanismos de soma de verificação (*checksum*, para detecção de erros), retransmissão de pacotes perdidos ou corrompidos, confirmações cumulativas e seletivas e temporizadores. Nesta seção, discutimos primeiramente os serviços fornecidos pelo TCP; em seguida, as características do TCP em mais detalhes. O TCP é o protocolo mais comum da camada de transporte na Internet.

3.4.1 Serviços do TCP

Antes de discutirmos o TCP em detalhes, explicaremos os serviços oferecidos pelo TCP para os processos da camada de aplicação.

Comunicação processo a processo

Assim como acontece com o UDP, o TCP fornece comunicação processo a processo usando números de porta. Já listamos alguns dos números de portas usados pelo TCP na Tabela 3.1 da seção anterior.

Serviço de encaminhamento de fluxos

O TCP, ao contrário do UDP, é um protocolo orientado a fluxo. No UDP, um processo envia mensagens com limites predefinidos para a entrega pela camada UDP. O UDP adiciona seu próprio cabeçalho a cada uma dessas mensagens e as entrega para a camada IP para a transmissão. Cada mensagem do processo é chamada *datagrama de usuário* e acaba se tornando um datagrama IP. Nem o IP nem o UDP reconhecem qualquer relação entre os datagramas.

O TCP, por outro lado, permite que o processo emissor entregue dados como um fluxo de *bytes* e permite que o processo receptor obtenha os dados como um fluxo de *bytes*. O TCP cria um ambiente no qual os dois processos parecem estar conectados por um "tubo" imaginário, que transporta os seus *bytes* através da Internet. Esse ambiente imaginário é ilustrado na Figura 3.41. O processo emissor produz o fluxo (escreve dados nele) e o processo receptor consome o fluxo (lê dados dele).

Figura 3.41 Entrega de fluxo.

Buffers de envio e de recepção

Como os processos de envio e de recepção não precisam necessariamente escrever ou ler dados na mesma velocidade, o TCP precisa de *buffers* para armazenamento temporário. Existem dois *buffers*, o de envio e o de recepção, um para cada direção da comunicação. Veremos mais adiante que esses *buffers* são também necessários para os mecanismos de controle de fluxo e de erros utilizados pelo TCP. Uma forma de implementar um *buffer* é utilizando um vetor circular de posições de 1 *byte*, conforme mostra a Figura 3.42. Por simplicidade, mostramos dois *buffers* de 20 *bytes* cada; normalmente, os *buffers* apresentam centenas ou milhares de *bytes*, dependendo da aplicação. Também mostramos os *buffers* com o mesmo tamanho, o que nem sempre é o caso.

A figura mostra o movimento dos dados em uma direção. No emissor, o *buffer* tem três tipos de espaços. A seção branca contém espaços vazios que podem ser preenchidos pelo processo emissor (produtor). A área em cinza contém os *bytes* que foram enviados, mas ainda não foram confirmados. O emissor TCP mantém esses *bytes* no *buffer* até que ele receba uma confirmação. A área sombreada contém os *bytes* a serem enviados pelo emissor TCP. No entanto, conforme veremos mais adiante neste capítulo, o TCP pode ser capaz de enviar apenas uma parte dessa seção sombreada. Isto pode ocorrer devido à lentidão do processo receptor ou ao congestionamento da rede. Também perceba que após os *bytes* nos espaços cinzas serem confirmados, os espaços são reutilizados e disponibilizados para o uso pelo processo emissor. É por isso que mostramos um *buffer* circular.

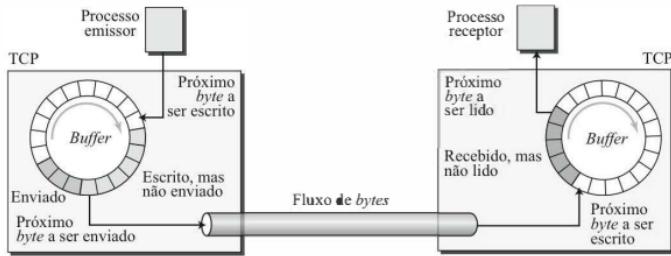



Figura 3.42 *Buffers* de envio e de recepção.

A operação do *buffer* no receptor é mais simples. O *buffer* circular é dividido em duas áreas (mostradas como branca e cinza). A área branca contém espaços vazios a serem preenchidos por *bytes* recebidos da rede. As seções cinzas contêm *bytes* recebidos que podem ser lidos pelo processo receptor. Quando um *byte* é lido pelo processo receptor, o espaço é reutilizado e adicionado ao conjunto de espaços vazios.

Segmentos

Embora o *buffer* lide com a disparidade entre as velocidades dos processos de produção e de consumo, precisamos de mais uma etapa antes que possamos enviar os dados. A camada de rede, como uma prestadora de serviços para o TCP, precisa enviar os dados em pacotes e não como um fluxo de *bytes*. Na camada de transporte, o TCP agrupa um número de *bytes* para formar um pacote denominado um *segmento*. O TCP adiciona um cabeçalho a cada segmento (para fins de controle) e entrega o segmento para a transmissão pela camada de rede. Os segmentos são encapsulados em um datagrama IP e transmitidos. Essa operação inteira é transparente para o processo de recepção. Mais adiante, veremos que os segmentos podem ser recebidos fora de ordem, perdidos ou corrompidos e reenviados. Tudo isto é tratado pelo receptor TCP, sem que o processo receptor na camada de aplicação tome ciência das atividades do TCP. A Figura 3.43 mostra como os segmentos são criados a partir dos *bytes* nos *buffers*.

Note que os segmentos não são necessariamente todos do mesmo tamanho. Na figura, para simplificar, mostramos um segmento que transporta 3 *bytes* e outro que transporta 5 *bytes*. Na realidade, os segmentos transportam centenas, senão milhares, de *bytes*.

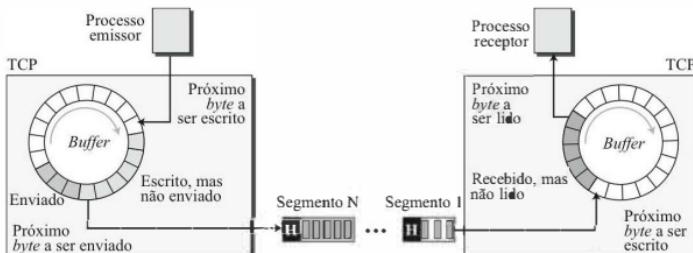



Figura 3.43 Segmentos TCP.

Comunicação full-duplex

O TCP fornece um *serviço full-duplex*, no qual os dados podem fluir em ambas as direções ao mesmo tempo. Cada extremidade do TCP tem, portanto, seu próprio *buffer* de envio e de recepção e os segmentos se movem em ambas as direções.

Multiplexação e demultiplexação

Como o UDP, o TCP executa a multiplexação no emissor e a demultiplexação no receptor. No entanto, como o TCP é um protocolo orientado à conexão, uma conexão precisa ser estabelecida para cada par de processos.

Serviço orientado à conexão

O TCP, ao contrário do UDP, é um protocolo orientado à conexão. Quando um processo no local A quer enviar e receber dados de outro processo no local B, as três fases seguintes ocorrem:

1. As duas camadas TCP estabelecem uma conexão lógica entre elas.
2. Os dados são trocados em ambas as direções.
3. A conexão é finalizada.

Perceba que essa é uma conexão lógica, não física. O segmento TCP é encapsulado em um datagrama IP e pode ser enviado fora de ordem, perdido ou corrompido e depois reenviado. Cada segmento pode ser encaminhado por um caminho diferente até chegar ao destino. Não há uma conexão física. O TCP cria um ambiente orientado a fluxo, no qual aceita a responsabilidade de entregar os *bytes* em ordem para o outro lado da conexão.

Serviço confiável

O TCP é um protocolo de transporte confiável. Ele usa um mecanismo de confirmações para verificar se os dados chegaram sãos e salvos no destino. Discutimos esse recurso mais adiante na seção de controle de erros.

3.4.2 Características do TCP

Para fornecer os serviços mencionados na seção anterior, o TCP apresenta várias características que são brevemente resumidas nesta seção e discutidas em detalhes mais adiante.

Sistema de numeração

Embora o *software* TCP mantenha um registro dos segmentos sendo transmitidos ou recebidos, não existe um campo para o valor do número de segmento no cabeçalho do segmento. Em vez disso, há dois campos, chamados *número de sequência* e *número de confirmação*, que referem-se a um número de *byte* e não de segmento.

Número de *byte*

O TCP numera todos os *bytes* de dados (octetos) que são transmitidos em uma conexão. A numeração é independente em cada direção. Quando o TCP recebe *bytes* de dados vindos de um processo, o TCP os armazena no *buffer* de envio e os numera. A numeração não começa necessariamente a partir de 0. Na verdade, o TCP escolhe um número qualquer entre 0 e $2^{32} - 1$ para o primeiro *byte*. Por exemplo, se o número for 1.057 e a quantidade total de dados a ser enviada for de 6.000 *bytes*,

os *bytes* são numerados de 1.057 a 7.056. Veremos que a numeração de *bytes* é usada para o controle de fluxo e de erros.

Os *bytes* dos dados a serem transferidos em cada conexão são numerados pelo TCP. A numeração começa com um número gerado arbitrariamente.

Número de sequência

Após os *bytes* terem sido numerados, o TCP atribui um número de sequência para cada segmento que está sendo enviado. O número de sequência, em cada direção, é definido da seguinte forma:

1. O número de sequência do primeiro segmento é o Número de Sequência Inicial (ISN – Initial Sequence Number), que é um número aleatório.
2. O número de sequência de qualquer outro segmento é o número de sequência do segmento anterior mais o número de *bytes* (reais ou imaginários) transportado pelo segmento anterior. Mais adiante, mostramos que alguns segmentos de controle são vistos como se transportassem um *byte* imaginário.

Exemplo 3.17

Considere que uma conexão TCP esteja transferindo um arquivo de 5.000 *bytes*. O primeiro é numerado como 10.001. Quais são os números de sequência para cada segmento se os dados forem enviados em cinco segmentos, cada um com 1.000 *bytes*?

Solução

A seguir, são mostrados os números de sequência para cada segmento:

Segmento 1	→	Número de Sequência:	10.001	**Faixa:**	10.001	a	11.000
Segmento 2	→	Número de Sequência:	11.001	**Faixa:**	11.001	a	12.000
Segmento 3	→	Número de Sequência:	12.001	**Faixa:**	12.001	a	13.000
Segmento 4	→	Número de Sequência:	13.001	**Faixa:**	13.001	a	14.000
Segmento 5	→	Número de Sequência:	14.001	**Faixa:**	14.001	a	15.000

O valor no campo de número de sequência de um segmento define o número atribuído ao primeiro *byte* de dados contido naquele segmento.

Quando um segmento carrega uma combinação de dados e informações de controle (mecanismo de carona, ou *piggybacking*), ele usa um número de sequência. Se um segmento não transporta dados do usuário, ele não define logicamente um número de sequência. O campo está lá, mas seu valor não é válido. No entanto, alguns segmentos, quando carregam apenas informação de controle, precisam de um número de sequência para possibilitar uma confirmação do receptor. Esses segmentos são utilizados para o estabelecimento, finalização ou cancelamento da conexão. Cada um deles consome um número de sequência, como se carregasse um *byte*, mas não existem dados reais transportados. Entraremos em mais detalhes sobre essa questão quando discutirmos conexões.

Número de confirmação

Conforme discutimos anteriormente, a comunicação no TCP é *full-duplex*; quando uma conexão é estabelecida, ambas as partes podem enviar e receber dados ao mesmo tempo. Cada lado numera os

seus *bytes*, geralmente com um número de *byte* inicial diferente. O número de sequência em cada direção mostra o número do primeiro *byte* transportado pelo segmento. Cada lado também usa um número de confirmação para confirmar os *bytes* que recebeu, mas o número de confirmação define o número do próximo *byte* que o lado espera receber. Além disso, o número de confirmação é cumulativo, o que significa que cada lado toma o número do último *byte* que recebeu corretamente, soma 1 a esse valor, e depois transmite o resultado da soma como o número de confirmação. O termo *cumulativo* significa aqui que se um lado usa 5643 como um número de confirmação, ele recebeu todos os *bytes* entre o *byte* inicial e o 5642. Note que isto não significa que foram recebidos 5.642 *bytes*, pois o número do primeiro *byte* não necessariamente é 0.

O valor do campo de confirmação em um segmento define o número do próximo *byte* que um lado espera receber. O número de confirmação é cumulativo.

3.4.3 Segmento

Antes de discutirmos o TCP em mais detalhes, explicaremos os pacotes TCP em si. Um pacote no TCP é denominado um **segmento**.

Formato

O formato de um segmento é mostrado na Figura 3.44. O segmento é formado por um cabeçalho de 20 a 60 *bytes*, seguido pelos dados do programa da camada de aplicação. O cabeçalho tem 20 *bytes* se não houver opções e pode chegar a ter 60 *bytes* se contiver opções. Discutimos alguns dos campos do cabeçalho nesta seção. O significado e a finalidade deles ficarão mais claros à medida que prosseguimos ao longo da seção.

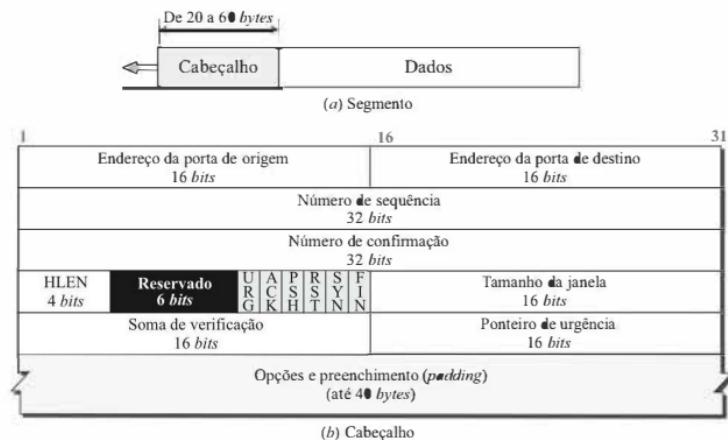



Figura 3.44 Formato do segmento TCP.

- **Endereço da porta de origem.** É um campo de 16 *bits* que define o número de porta do aplicativo na estação que está enviando o segmento.

- **Endereço da porta de destino.** É um campo de 16 *bits* que define o número de porta do aplicativo na estação que está recebendo o segmento.
- **Número de sequência.** Campo de 32 *bits* que define o número atribuído ao primeiro *byte* dos dados contidos nesse segmento. Como dissemos anteriormente, o TCP é um protocolo de transporte orientado a fluxo. Para garantir a conectividade, cada *byte* a ser transmitido é numerado. O número de sequência informa ao destino qual *byte* nessa sequência é o primeiro *byte* no segmento. Durante o estabelecimento da conexão (discutido mais adiante) cada parte usa um gerador de números aleatórios para criar um **Número de Sequência Inicial** (ISN – Initial Sequence Number), que normalmente é diferente em cada direção.
- **Número de confirmação.** Campo de 32 *bits* que define o número de *bytes* que o receptor do segmento espera receber da outra parte. Se o receptor do segmento receber com sucesso o *byte* de número x da outra parte, ele retorna $x + 1$ como o número de confirmação. A confirmação e os dados podem ser enviados juntos, usando o mecanismo de carona (*piggybacking*).
- **Comprimento do cabeçalho (HLEN).** Campo de 4 *bits* que indica o número de palavras de 4 *bytes* no cabeçalho TCP. O comprimento do cabeçalho pode valer entre 20 e 60 *bytes*. Portanto, o valor deste campo fica sempre entre 5 ($5 \times 4 = 20$) e 15 ($15 \times 4 = 60$).
- **Controle.** Campo que define 6 diferentes *bits* de controle ou marcadores, conforme mostra a Figura 3.45. Um ou mais desses *bits* pode ser ativado de cada vez. Eles permitem o controle de fluxo, o estabelecimento e a finalização da conexão, o cancelamento da conexão e a definição do modo de transferência de dados no TCP. Uma breve descrição de cada *bit* é mostrada na figura. Vamos discuti-los mais a fundo quando estudarmos a operação detalhada do TCP mais adiante neste capítulo.

URG: Ponteiro de urgência é válido
ACK: Confirmação é válida
PSH: Pedido de *push* (empurrar)
RST: Reiniciar a conexão
SYN: Sincronizar números de sequência
FIN: Finalizar a conexão

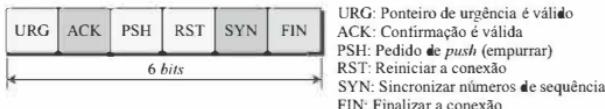



Figura 3.45 Campo de controle.

- **Tamanho da janela.** Campo que define o tamanho da janela do TCP para o envio de *bytes*. Perceba que o comprimento desse campo é de 16 *bits*, o que significa que o tamanho máximo da janela é 65.535 *bytes*, valor normalmente conhecido como janela de recepção (*receiving window – rwnd*) e determinado pelo receptor. O emissor deve respeitar o valor ditado pelo receptor nesse caso.
- **Soma de verificação.** Campo de 16 *bits* que contém a soma de verificação (*checksum*). O cálculo da soma de verificação para o TCP segue o mesmo procedimento descrito para o UDP. No entanto, a utilização da soma de verificação no datagrama UDP é opcional, enquanto a da soma de verificação no TCP é obrigatória. O mesmo pseudocabeçalho, servindo ao mesmo propósito, é adicionado ao segmento. Para o pseudocabeçalho do TCP, o valor do campo de protocolo é 6. Ver a Figura 3.46.

O uso da soma de verificação (*checksum*) no TCP é obrigatório.

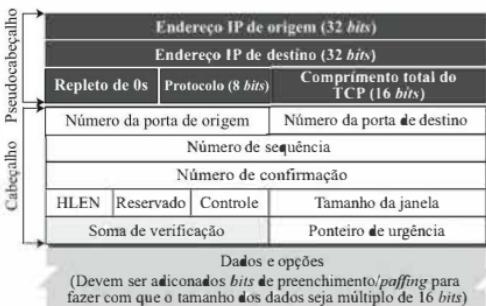



Figura 3.46 Pseudocabeçalho adicionado ao datagrama TCP.

- **Ponteiro de urgência.** Campo de 16 *bits*, que só é válido se o marcador de urgência valer 1, usado quando o segmento contém dados urgentes. Ele define um valor que deve ser adicionado ao número de sequência para obter o número do último *byte* urgente na seção de dados do segmento. Isto será discutido mais adiante neste capítulo.
- **Opções.** Pode haver até 40 *bytes* de informação opcional no cabeçalho TCP. Discutiremos algumas das opções usadas no cabeçalho TCP mais adiante nesta seção.

Encapsulamento

Um segmento TCP encapsula os dados recebidos da camada de aplicação; ele é encapsulado em um datagrama IP, que por sua vez é encapsulado em um quadro na camada de enlace de dados.

3.4.4 Conexão TCP

O TCP é orientado à conexão. Conforme discutido anteriormente, um protocolo de transporte orientado à conexão estabelece um caminho lógico entre a origem e o destino. Todos os segmentos pertencentes a uma mensagem são, então, enviados por meio desse caminho lógico. O uso de um único caminho lógico para toda a mensagem facilita o processo de confirmação, bem como a retransmissão de quadros danificados ou perdidos. Você pode se perguntar como o TCP, que utiliza os serviços do IP, um protocolo não orientado à conexão, pode ser orientado à conexão. O ponto é que uma conexão TCP é lógica, não física. O TCP opera em um nível mais elevado. O TCP utiliza os serviços do IP para entregar segmentos individuais para o receptor, mas ele controla a conexão em si. Se um segmento for perdido ou danificado, ele é retransmitido. Ao contrário do TCP, o IP não toma ciência dessa retransmissão. Se um segmento chegar fora de ordem, o TCP o armazena até que os segmentos faltantes cheguem; o IP não toma ciência desse reordenamento.

No TCP, a transmissão orientada à conexão exige três passos: estabelecimento de conexão, transferência de dados e finalização de conexão.

Estabelecimento de conexão

O TCP transmite dados em modo *full-duplex*. Quando dois TCPs em duas máquinas estão conectados, eles conseguem enviar segmentos um para o outro ao mesmo tempo. Assim, cada parte deve iniciar a comunicação e obter a aprovação da outra antes que os dados sejam transferidos.

Apresentação em três vias

O estabelecimento da conexão TCP é denominado **apresentação em três vias** (three-way handshaking). No nosso exemplo, um aplicativo, chamado cliente, quer criar uma conexão com outro aplicativo, chamado servidor, usando o TCP como o protocolo da camada de transporte.

O processo se inicia com o servidor. O programa-servidor informa seu TCP que ele está pronto para aceitar uma conexão. Essa solicitação é chamada *abertura passiva*. Embora o TCP servidor esteja pronto para aceitar uma conexão de qualquer máquina no mundo, ele não pode criar a conexão sozinho.

O programa-cliente envia uma solicitação para uma *abertura ativa*. Um cliente que deseja se conectar a um servidor aberto informa o seu TCP para que ele se conecte a um servidor específico. O TCP pode, então, começar o processo de *three-way handshaking*, conforme mostra a Figura 3.47.

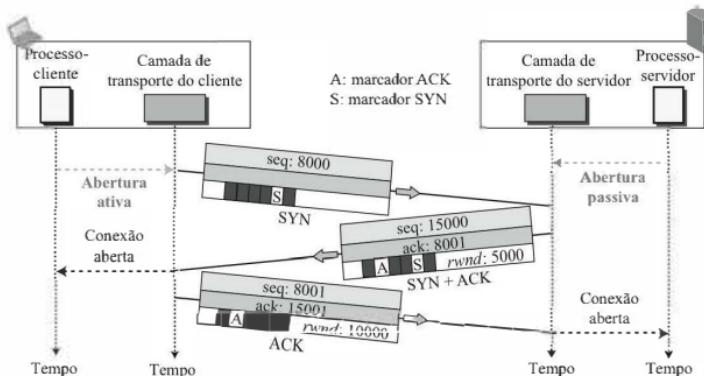



Figura 3.47 Estabelecimento de conexão usando o *three-way handshaking* (apresentação em três vias).

Para mostrar o processo, usamos linhas de tempo. Cada segmento tem valores para todos os seus campos de cabeçalho e, possivelmente, também para alguns dos campos opcionais. No entanto, mostramos apenas os poucos campos necessários para entender cada fase. Apresentamos o número de sequência, o número de confirmação, os marcadores de controle (apenas aqueles que estão ativados) e o tamanho da janela quando relevante. Os três passos dessa fase são os seguintes.

1. O cliente envia o primeiro segmento, de SYN, que só tem o marcador SYN ativado. Esse segmento é usado para sincronizar os números de sequência. O cliente, no nosso exemplo, escolhe um número aleatório como o primeiro número de sequência e o envia para o servidor. Esse número de sequência é chamado Número de Sequência Inicial (ISN – Initial Sequence Number). Note que o segmento não contém um número de confirmação e também não especifica o tamanho da janela; uma definição do tamanho da janela só faz sentido quando um segmento incluir uma confirmação. O segmento também pode incluir algumas opções que discutimos mais adiante neste capítulo. Perceba que o segmento de SYN é um segmento de controle e não transporta qualquer dado. No entanto, ele consome um número de sequência porque precisa ser confirmado. Podemos dizer que o segmento SYN carrega um *byte* imaginário.

Um segmento de SYN não pode transportar dados, mas consome um número de sequência.

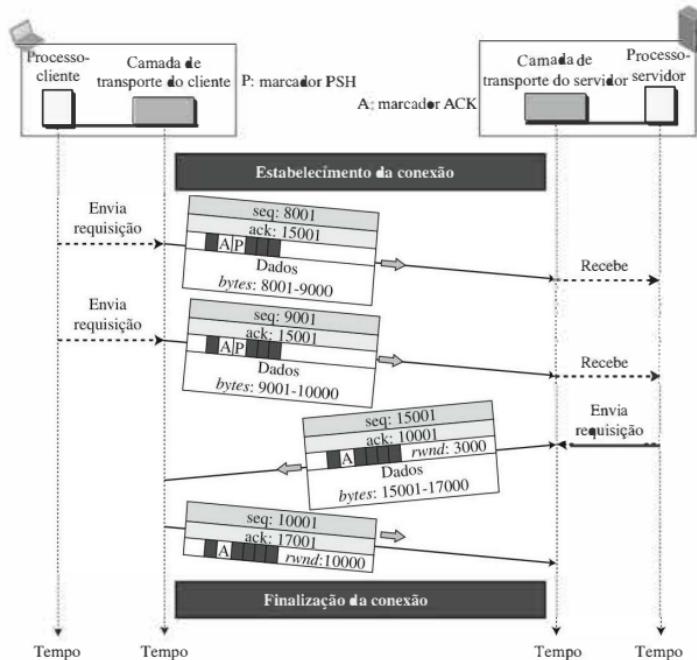
2. O servidor envia o segundo segmento, de SYN + ACK com dois *bits* de marcadores ativados: SYN e ACK. Esse segmento tem dupla finalidade. Primeiramente, atua com um segmento de SYN para a comunicação na outra direção. O servidor usa esse segmento para inicializar um número de sequência para numerar os *bytes* enviados do servidor para o cliente. O servidor também confirma o recebimento do segmento de SYN do cliente, ativando o marcador de ACK e inserindo o próximo número de sequência que ele espera receber do cliente. Já que esse segmento contém uma confirmação, ele também precisa definir o tamanho da janela de recepção, *rwnd* (para ser usado pelo cliente), conforme veremos na seção sobre controle de fluxo. Como esse segmento está desempenhando o papel de um de SYN, precisa ser confirmado. Ele consome, portanto, um número de sequência.

Um segmento de SYN + ACK não pode transportar dados, mas consome um número de sequência.

3. O cliente envia o terceiro segmento. É apenas um segmento de ACK. Ele confirma o recebimento do segundo segmento usando o marcador de ACK e o campo de número de confirmação. Perceba que o segmento de ACK não consome qualquer número de sequência se não transportar dados. Contudo, algumas implementações permitem que o terceiro segmento transporte o primeiro bloco de dados do cliente na fase de conexão. Nesse caso, o segmento consome uma quantidade de números de sequência correspondente ao número de *bytes* de dados.

Um segmento de ACK, se não estiver transportando dados, não consome um número de sequência.

Ataque de inundação de SYN

O processo de estabelecimento de conexão no TCP é suscetível a um problema de segurança grave denominado **ataque de inundação de SYN** (*SYN flooding attack*). Isto acontece quando um ou mais atacantes maliciosos enviam um grande número de segmentos de SYN para um servidor como se cada um deles fosse proveniente de um cliente diferente, algo feito falsificando os endereços IP de origem nos datagramas. O servidor, considerando que os clientes estão enviando uma abertura ativa, aloca os recursos necessários, por exemplo, criando tabelas para o bloco de transferência de controle (TCB – Transfer Control Block) e inicializando temporizadores. O servidor TCP envia, então, segmentos de SYN + ACK para os clientes falsos, que são perdidos. Enquanto o servidor espera pela terceira etapa do processo de *handshaking*, no entanto, os recursos ficam alocados sem serem utilizados. Se, durante esse curto período, o número de segmentos de SYN for grande, então, o servidor pode ficar sem recursos e ser incapaz de aceitar pedidos de clientes válidos. Esse ataque de inundação de SYN pertence a um grupo de ataques contra a segurança, conhecido como **ataque de negação de serviço**, no qual um atacante monopoliza um sistema com tantos pedidos de serviço que o sistema fica sobrecarregado e passa a negar a prestação de serviço aos pedidos válidos.

Algumas implementações do TCP têm estratégias para atenuar o efeito de um ataque de SYN. Algumas impõem um limite no número de pedidos de conexão durante um determinado período. Outras tentam filtrar os datagramas provenientes de endereços de origem indesejados. Uma estratégia recente é adiar a alocação de recursos até que o servidor possa verificar se o pedido de conexão é proveniente de um endereço IP válido, usando algo chamado ***cookie****. O SCTP, um protocolo da camada de transporte mais recente, sobre o qual discutimos no Capítulo 8, usa essa estratégia.

* N. de T.: Literalmente, a palavra *cookie* significa "biscoito" em inglês. Um *cookie* é um pequeno arquivo que contém informações sobre o cliente que acessa o serviço (por exemplo, um identificador daquele cliente).

Transferência de dados

Após a conexão ser estabelecida, a transferência bidirecional de dados pode acontecer. O cliente e o servidor podem enviar dados e confirmações em ambos os sentidos. Estudaremos as regras de confirmação no fim do capítulo; por ora, basta saber que os dados trafegando na mesma direção de uma confirmação são transportados usando o mesmo segmento. A confirmação é transportada junto com os dados (usando o mecanismo de carona, ou *piggybacking*). A Figura 3.48 mostra um exemplo.

Nesse exemplo, depois que uma conexão é estabelecida, o cliente envia 2.000 *bytes* de dados em dois segmentos. O servidor envia, então, 2.000 *bytes* em um segmento. O cliente envia mais um segmento. Os três primeiros segmentos transportam dados e confirmações, mas o último segmento tem apenas uma confirmação porque não há mais dados a serem enviados. Observe os valores dos números de sequência e de confirmação. Os segmentos de dados enviados pelo cliente têm o marcador PSH (empurrar) ativado, de forma que o servidor TCP sabe que ele deve enviar os dados para o processo servidor assim que eles forem recebidos. Discutimos o uso desse marcador com mais detalhes mais adiante. O segmento vindo do servidor, por outro lado, não tem o marcador de PSH ativado. A maioria das implementações do TCP tem a opção de ativar ou não esse marcador.

Figura 3.48 Transferência de dados.

Empurrando dados

Vimos que o TCP emissor usa um *buffer* para armazenar o fluxo de dados proveniente do aplicativo emissor. O TCP emissor pode selecionar o tamanho do segmento. O TCP receptor também armazena os dados em um *buffer* quando eles chegam e os entrega ao aplicativo receptor quando tal programa estiver pronto ou quando for conveniente para o TCP receptor. Esse tipo de flexibilidade aumenta a eficiência do TCP.

Entretanto, há ocasiões em que o aplicativo não precisa de tal flexibilidade. Por exemplo, considere um aplicativo que se comunica interativamente com outro na outra extremidade da conexão. O aplicativo em um local quer enviar um bloco de dados para o aplicativo em outro local e receber uma resposta imediata. Atrasos na transmissão e na entrega dos dados podem não ser aceitáveis pelo aplicativo.

O TCP pode lidar com tal situação. O aplicativo no emissor pode solicitar uma operação de "empurrar". Isto significa que o TCP emissor não deve aguardar a janela ser preenchida. Deve criar um segmento e enviá-lo imediatamente. O TCP emissor também deve marcar o *bit* "empurrar" (PSH) para que o TCP receptor saiba que o segmento contém dados que devem ser entregues ao aplicativo receptor o mais rápido possível, não devendo esperar por mais dados, o que significa transformar o TCP orientado a *byte* em um TCP orientado a bloco, mas o TCP pode escolher se quer ou não usar esse recurso.

Dados urgentes

O TCP é um protocolo orientado a fluxo, ou seja, os dados são passados pelo aplicativo para o TCP como um fluxo de *bytes*. Cada *byte* de dados tem uma posição no fluxo. No entanto, há ocasiões em que um aplicativo precisa enviar *bytes urgentes*, alguns *bytes* que precisam ser tratados de uma maneira especial pela aplicação na outra extremidade. A solução é enviar um segmento com o *bit* URG ativado. O aplicativo emissor informa ao TCP emissor que o bloco de dados é urgente. O TCP emissor cria um segmento e insere os dados urgentes no início do segmento; o restante pode conter dados normais obtidos do *buffer*. O campo de ponteiro de urgência no cabeçalho especifica o fim dos dados urgentes (o último *byte* de dados urgentes). Por exemplo, se o número de sequência do segmento for 15.000 e o valor do ponteiro de urgência for 200, o primeiro *byte* de dados urgentes é o *byte* 15.000 e o último *byte* é o 15.200. O restante dos *bytes* no segmento (se existirem) não são urgentes.

É importante mencionar que a urgência de dados no TCP não leva a um serviço prioritário nem a um serviço de dados fora da banda (*out-of-band*)*, como algumas pessoas podem pensar. Na realidade, o modo urgente do TCP é um serviço pelo qual o aplicativo no lado do emissor marca alguma porção do fluxo de *bytes* como algo que requer tratamento especial pelo aplicativo no lado do receptor. O TCP receptor entrega os *bytes* (urgentes ou não urgentes) para o aplicativo de forma ordenada, mas informa ao aplicativo sobre o início e o fim dos dados urgentes. Fica a cargo do aplicativo decidir o que fazer com os dados urgentes.

Finalização da conexão

Qualquer uma das duas partes envolvidas na troca de dados (cliente ou servidor) pode fechar a conexão, embora geralmente isso seja iniciado pelo cliente. A maioria das implementações atuais permite duas opções para o término de conexão: *three-way handshaking* (apresentação em três vias) e *four-way handshaking* (apresentação em quatro vias), com uma opção de semifechamento.

Apresentação em três vias

A maioria das implementações atuais suporta o *three-way handshaking*, ou apresentação em três vias, para a finalização de conexão, conforme mostra a Figura 3.49.

* N. de T.: Dados fora da banda referem-se à transmissão de alguns sinais (informações de controle, por exemplo) em frequências distintas da largura de banda disponível para a transferência de voz ou dados de um canal de comunicação.

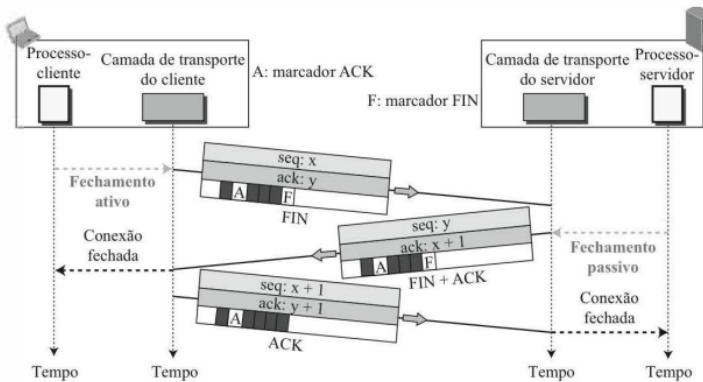



Figura 3.49 Finalização da conexão usando *three-way handshaking*.

1. Nessa situação, o TCP cliente, após receber um comando de fechamento do processo--cliente, envia o primeiro segmento, de FIN, no qual o marcador FIN vale 1. Observe que um segmento de FIN pode incluir o último bloco de dados enviado pelo cliente ou pode ser apenas um segmento de controle, conforme mostrado na figura. Se for apenas um segmento de controle, ele consome somente um número de sequência porque precisa ser confirmado.

O segmento de FIN consome apenas um número de sequência se não transportar dados.

2. O TCP servidor, após receber o segmento de FIN, informa seu processo dessa situação e envia o segundo segmento, de FIN+ACK, para confirmar a recepção do segmento de FIN vindo do cliente e, ao mesmo tempo, para anunciar o fechamento da conexão na outra direção. Esse segmento também pode conter o último bloco de dados do servidor. Se ele não transportar dados, consome apenas um número de sequência, pois precisa ser confirmado.

O segmento de FIN + ACK consome apenas um número de sequência se não transportar dados.

3. O TCP cliente envia o último segmento, de ACK, para confirmar a recepção do segmento de FIN proveniente do servidor TCP. Esse segmento contém o número de confirmação, que vale um mais o número de sequência recebido no segmento de FIN vindo do servidor. Ele não pode carregar dados e não consome números de sequência.

Semifechamento

No TCP, uma extremidade pode parar de enviar dados enquanto continua a recebê-los. Isto é chamado estado **semifechado** (*half-closed*). O servidor ou o cliente pode enviar uma solicitação de semifechamento, e pode ocorrer quando o servidor precisa de todos os dados antes que o processamento possa começar. Um bom exemplo é uma operação de ordenação. Quando o cliente envia dados para

o servidor para que eles sejam ordenados, o servidor precisa receber todos os dados antes de começar o processo de ordenação. Ou seja, o cliente, depois de enviar todos os dados, pode fechar a conexão no sentido do cliente para o servidor. Entretanto, a conexão na direção do servidor para o cliente deve permanecer aberta para que os dados ordenados possam ser retornados. O servidor, após receber os dados, ainda precisa de tempo para realizar a ordenação; a sua conexão na direção de saída deve permanecer aberta. A Figura 3.50 mostra um exemplo de uma conexão semifechada.

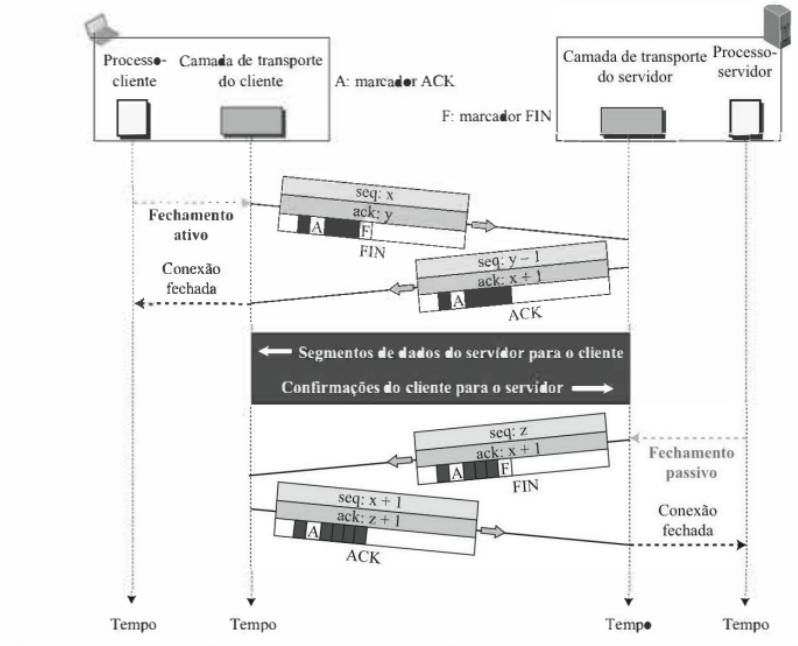



Figura 3.50 Semifechado

A transferência de dados do cliente para o servidor é interrompida. O cliente semifecha a conexão enviando um segmento de FIN. O servidor aceita o semifechamento enviando um segmento de ACK, mas ainda pode enviar dados. Quando o servidor tiver enviado todos os dados processados, ele envia um segmento de FIN que é confirmado por um ACK do cliente.

Depois de semifechar a conexão, os dados podem trafegar do servidor para o cliente e as confirmações podem trafegar do cliente para o servidor. O cliente não pode mais enviar dados para o servidor.

Reiniciar conexão

O TCP em uma das extremidades pode recusar uma solicitação de conexão, abortar uma conexão existente ou finalizar uma ociosa. Tudo isso é feito com o marcador RST (*reset*).

3.4.5 Diagrama de transição de estados

Para manter o controle sobre todos os diferentes eventos que acontecem durante o estabelecimento da conexão, a finalização da conexão e a transferência de dados, o TCP foi projetado como uma máquina de estados finitos (MEF), conforme ilustrado na Figura 3.51.

A figura mostra as duas MEFs usadas pelo cliente e pelo servidor TCP combinadas em um só diagrama. Os retângulos de cantos arredondados representam os estados. A transição de um estado para outro é mostrada usando setas. Cada seta tem duas cadeias de caracteres separadas por uma barra. A primeira cadeia de caracteres corresponde à entrada que o TCP recebe. A segunda corresponde à saída que o TCP envia. As setas pretas tracejadas na figura representam a transição pela qual um servidor normalmente passa; as setas pretas sólidas mostram as transições pelas quais um cliente normalmente passa. No entanto, em algumas situações, um servidor realiza transições por uma linha sólida ou um cliente realiza transições por uma linha tracejada. As linhas cinzas mostram situações especiais. Perceba que o retângulo de canto arredondado marcado como ESTABELECIDO representa, na verdade, dois conjuntos de estados, um conjunto para o cliente e outro para o servidor, que são utilizados para o controle de fluxo e de erros, conforme explicado mais adiante. Discutimos alguns temporizadores mencionados na figura, incluindo o temporizador 2MSL, no fim do capítulo. Usamos diversos cenários baseados na Figura 3.51 e mostramos parte da figura em cada caso.

O estado marcado como ESTABELECIDO na MEF representa, na verdade, dois diferentes conjuntos de estados pelos quais o cliente e o servidor passam para transferir dados.

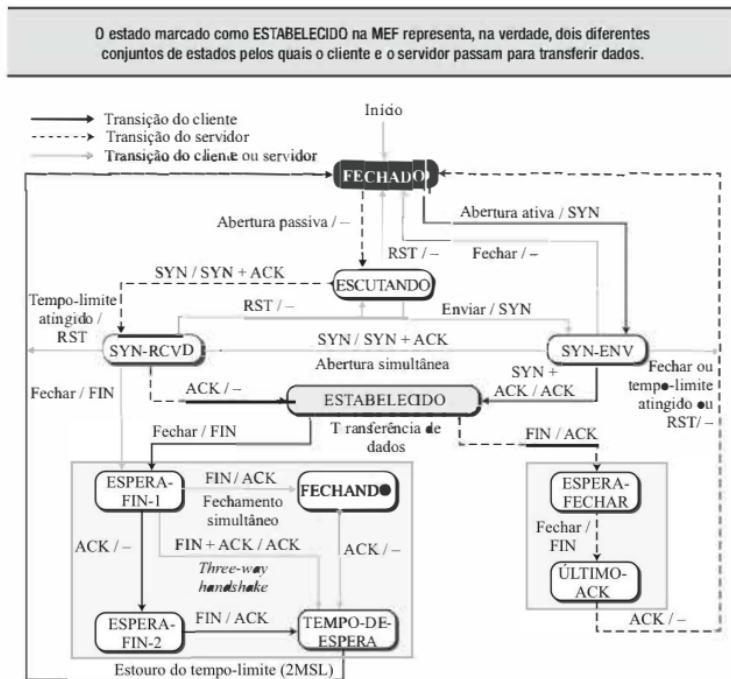



Figura 3.51 Diagrama de transição de estados.

A Tabela 3.2 mostra a lista de estados para o TCP.

Tabela 3.2 Estados do TCP.

Estado	Descrição
FECHADO	Não há uma conexão
ESCUTANDO	Abertura passiva recebida; esperando por SYN
SYN-ENV	SYN enviado; esperando por ACK
SYN-RCVD	SYN+ACK enviado; esperando por ACK
ESTABELECIDO	Conexão estabelecida; transferência de dados em andamento
ESPERA-FIN-1	Primeiro FIN enviado; esperando por ACK
ESPERA-FIN-2	ACK para o primeiro FIN recebido; esperando por segundo FIN
ESPERA-FECHAR	Primeiro FIN recebido, ACK enviado; esperando pelo fechamento da aplicação
TEMPO-DE-ESPERA	Segundo FIN recebido, ACK enviado; esperando pela expiração do temporizador 2MSL
ÚLTIMO-ACK	Segundo FIN enviado; esperando por ACK
FECHANDO	Ambos os lados decidem fechar simultaneamente

Cenários

Para entender as máquinas de estado TCP e os diagramas de transição, discutimos um cenário ilustrativo nesta seção.

Um cenário de semifechamento

A Figura 3.52 mostra o diagrama de transição de estados desse cenário.

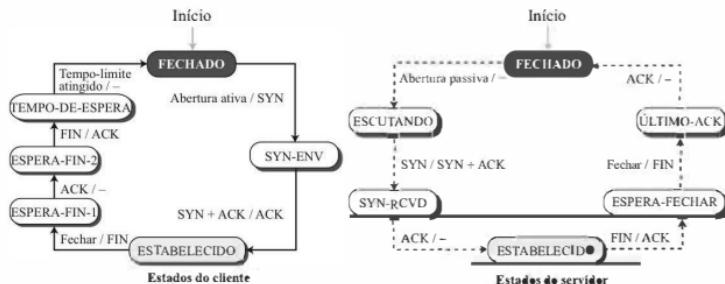



Figura 3.52 Diagrama de transição com a finalização de uma conexão semifechada.

O processo-cliente envia um comando de *abertura ativa* ao seu TCP para solicitar uma conexão com um endereço de *socket* específico. O TCP envia um segmento de SYN e passa para o estado **SYN-ENV**. Após receber um segmento de SYN + ACK, o TCP envia um segmento de ACK e vai para o estado **ESTABELECIDO**. Os dados são transferidos, possivelmente em ambas as direções, e a recepção é confirmada. Quando o processo-cliente não tem mais dados para enviar, ele envia um comando chamado *fechamento ativo*. O TCP envia um segmento de FIN e vai para o estado **ESPERA-FIN-1**. Quando ele recebe o segmento de ACK, ele vai para o estado **ESPERA-FIN-2**. Quando o cliente recebe o segmento de FIN, ele envia um segmento de ACK e vai para o estado **TEMPO-DE-ESPERA**. O cliente permanece nesse estado por 2 MSL segundos (ver temporizadores do TCP no fim do capítulo). Quando o temporizador correspondente expira, o cliente vai para o estado **FECHADO**.

O processo-servidor envia um comando de *abertura passiva*. O TCP servidor vai para o estado **ESCUTAR** e permanece passivamente nesse estado até receber um segmento de SYN. O TCP envia, então, um segmento de SYN + ACK e vai para o estado **SYN-RCBD**, no qual ele espera até que o cliente envie um segmento de ACK. Depois de receber o segmento de ACK, o TCP vai para o estado **ESTABELECIDO**, no qual podem ocorrer transferências de dados. O TCP permanece nesse estado até receber um segmento de FIN vindo do cliente, o que significa que não existem mais dados a serem transferidos e que a conexão pode ser fechada. O servidor, ao receber o segmento de FIN, envia para o servidor todos os dados na fila, juntamente com um marcador de EOF* virtual, o que significa que a conexão deve ser fechada. Ele envia um segmento de ACK e vai para o estado **ESPERA-FECHAR**, mas adia a confirmação do segmento de FIN recebido do cliente até que ele receba um comando de *fechamento passivo* do seu processo. Após receber o comando de fechamento passivo, o servidor envia um segmento de FIN ao cliente e vai para o estado **ÚLTIMO-ACK**, no qual espera pelo ACK final. Quando o segmento de ACK é recebido do cliente, o servidor vai para o estado **FECHADO**. A Figura 3.53 mostra o mesmo cenário com os estados sobre a linha do tempo.

3.4.6 Janelas no TCP

Antes de discutir a transferência de dados no TCP e questões como o controle de fluxo, de erros e de congestionamento, descrevemos as janelas utilizadas no TCP, que usa duas janelas (de envio e de recepção) para cada direção da transferência de dados, o que leva a quatro janelas para uma comunicação bidirecional. Para simplificar a discussão, fazemos uma suposição não realista de que a comunicação é apenas unidirecional (digamos, do cliente para servidor); os detalhes da comunicação bidirecional podem ser inferidos usando duas comunicações unidirecionais com o mecanismo de carona (*piggybacking*).

Janela de envio

A Figura 3.54 mostra um exemplo de uma janela de envio. O tamanho da janela é de 100 *bytes*, porém mais adiante veremos que o tamanho da janela de envio é ditado pelo receptor (controle de fluxo) e pelo congestionamento na rede subjacente (controle de congestionamento). A figura mostra como uma janela de envio *abre*, *fecha* ou *encolhe*.

A janela de envio do TCP é similar àquela utilizada no protocolo de Repetição Seletiva, mas com algumas diferenças:

1. Uma diferença é a natureza das entidades relacionadas com a janela. O tamanho da janela no protocolo RS refere-se ao número de pacotes, enquanto o tamanho da janela no TCP refere-se ao número de *bytes*. Embora a transmissão no TCP ocorra segmento por segmento, as variáveis que controlam a janela são expressadas em *bytes*.

* N. de T.: A sigla EOF significa *End of File*, ou "fim do arquivo". Trata-se de um indicador de que o conjunto de dados acabou.

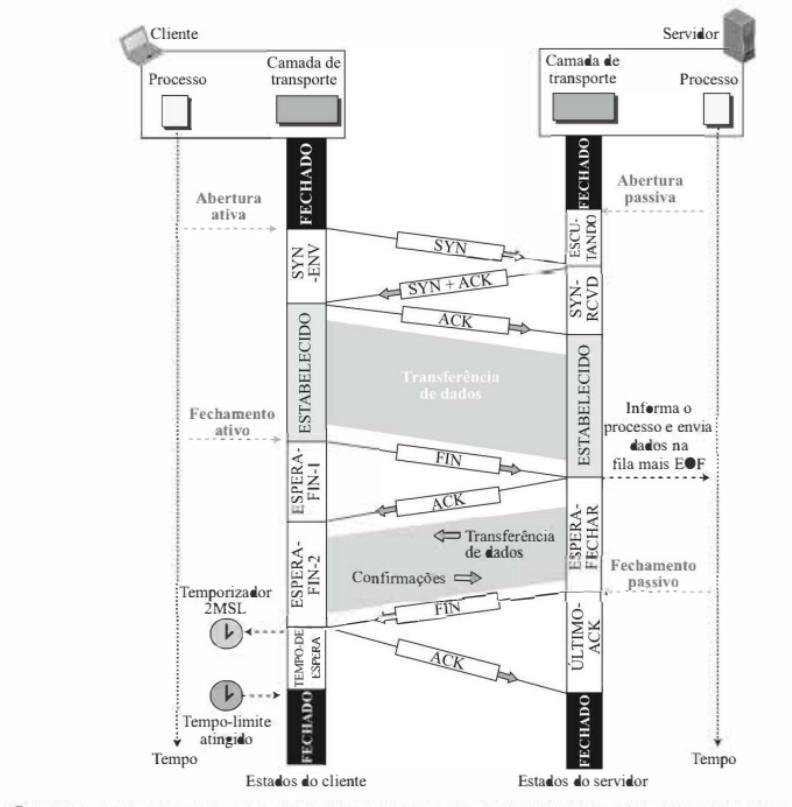



Figura 3.53 Diagrama de linha do tempo para o cenário comum.

2. A segunda diferença é que, em algumas implementações, o TCP pode armazenar dados recebidos do processo da camada de aplicação e enviá-los mais tarde, porém assume-se que o TCP emissor é capaz de enviar segmentos de dados tão logo ele os receba de seu processo.
3. Outra diferença é o número de temporizadores. Na teoria, o protocolo de Repetição Seletiva pode usar vários temporizadores para cada pacote enviado. Porém, conforme mencionado anteriormente, o protocolo TCP utiliza apenas um temporizador.

Janela de recepção

A Figura 3.55 mostra um exemplo de uma janela de recepção. O tamanho da janela é de 100 *bytes*. A figura também mostra como a janela de recepção abre e fecha; na prática, ela nunca deveria encolher.

Figura 3.54 Janela de envio no TCP.

Figura 3.55 Janela de recepção no TCP.

Há duas diferenças entre a janela de recepção no TCP e aquela usada pelo protocolo RS.

1. A primeira diferença é que o TCP permite que o processo receptor "puxe" os dados em seu próprio ritmo. Isto significa que parte do *buffer* alocado no receptor pode ser ocupada por *bytes* que foram recebidos e confirmados, mas estão aguardando para serem puxados pelo processo receptor. O tamanho da janela de recepção é sempre menor do que ou igual ao tamanho do *buffer*, conforme mostrado na Figura 3.55. Seu tamanho determina o número

de *bytes* provenientes do emissor que a janela de recepção pode aceitar antes de ficar sobrecarregada (controle de fluxo). Em outras palavras, o tamanho da janela de recepção, normalmente denotado por *rwnd (reception window)*, pode ser calculado como:

***rwnd* = tamanho do *buffer* – número de *bytes* aguardando para serem puxados**

2. A segunda diferença é a forma como as confirmações são utilizadas no protocolo TCP. Lembre-se de que a confirmação no protocolo RS é seletiva, indicando os pacotes não corrompidos que foram recebidos. O principal mecanismo de confirmação no TCP é a confirmação cumulativa, que anuncia o próximo *byte* que se espera receber (nesse sentido, o TCP é parecido com o GBN, discutido anteriormente). Porém, a nova versão do TCP usa tanto confirmações cumulativas quanto seletivas; discutimos essas abordagens no *site* do livro.

3.4.7 Controle de fluxo

Conforme discutido anteriormente, o *controle de fluxo* equilibra a taxa na qual um produtor cria dados com a taxa na qual um consumidor é capaz de utilizar os dados. O TCP separa o controle de fluxo do controle de erros. Nesta seção, discutimos o controle de fluxo, ignorando o controle de erros. Consideramos que o canal lógico entre o TCP emissor e o TCP receptor é livre de erros. A Figura 3.56 mostra a transferência de dados unidirecional entre um emissor e um receptor; o processo de transferência bidirecional de dados pode ser deduzido a partir do processo unidirecional.

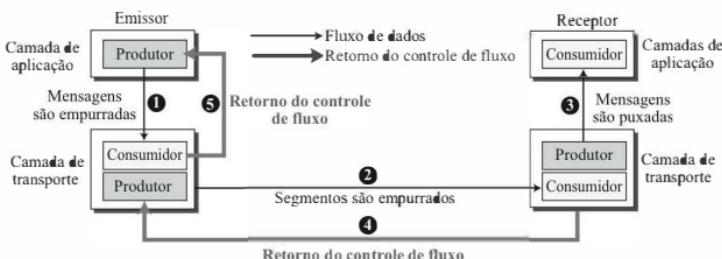



Figura 3.56 Fluxo de dados e retorno do controle de fluxo no TCP.

A figura mostra que os dados trafegam do processo emissor para o TCP emissor, do TCP emissor para o TCP receptor e do TCP receptor para o processo receptor (caminhos 1, 2 e 3). Retornos do controle de fluxo, entretanto, trafegam do TCP receptor para o TCP emissor e do TCP emissor para o processo emissor (caminhos 4 e 5). A maioria das implementações do TCP não fornece retorno do controle de fluxo do processo receptor para o TCP receptor; elas deixam o processo receptor puxar os dados do TCP receptor sempre que o processo estiver pronto para fazê-lo. Em outras palavras, o TCP receptor controla o TCP emissor; o TCP emissor controla o processo emissor.

O retorno do controle de fluxo do TCP emissor para o processo emissor (caminho 5) é feito por meio da simples rejeição dos dados pelo TCP emissor quando a sua janela estiver cheia. Isto significa que a nossa discussão sobre o controle de fluxo concentra-se no retorno enviado pelo TCP receptor para o TCP emissor (caminho 4).

Abrindo e fechando janelas

Para conseguir fazer o controle de fluxo, o TCP força o emissor e o receptor a ajustarem o tamanho das suas janelas, embora o tamanho do *buffer* em ambas as partes seja fixado quando a conexão é estabelecida. A janela de recepção fecha (move seu limite esquerdo para a direita), quando chegam mais *bytes* provenientes do emissor; ela abre (move seu limite direito mais para a direita), quando mais *bytes* são puxados pelo processo. Consideramos que ela não encolhe (o limite direito não se move para a esquerda).

A abertura, o fechamento e o encolhimento da janela de envio são controlados pelo receptor. A janela de envio fecha (move seu limite esquerdo para a direita), quando uma nova confirmação lhe permite fazê-lo. A janela de envio abre (move seu limite direito mais para a direita), quando o tamanho da janela (*rwnd*) anunciado pelo receptor lhe permite fazê-lo (quando novo nrAck + novo *rwnd* > último nrAck + último *rwnd*). A janela de envio encolhe caso essa situação não ocorra.

Um cenário

Mostramos como as janelas de envio e de recepção são definidas durante a fase de estabelecimento da conexão e como suas propriedades poderão mudar durante a transferência de dados. A Figura 3.57 mostra um exemplo simples de transferência de dados unidirecional (do cliente para o servidor). Por enquanto, ignoramos o controle de erros, considerando que nenhum segmento é corrompido,

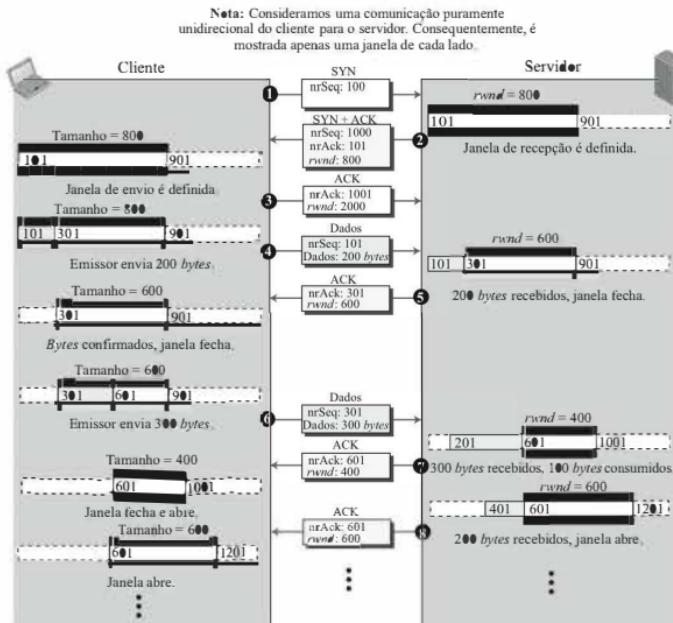
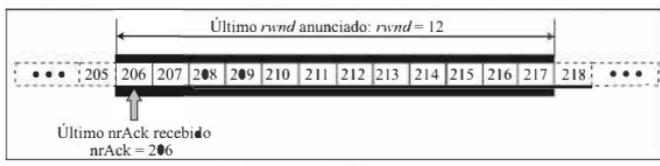



Figura 3.57 Um exemplo de controle de fluxo.

perdido, duplicado ou chega fora de ordem. Observe que mostramos apenas duas janelas para a transferência de dados unidirecional. Embora o cliente defina um tamanho da janela do servidor de 2000 no terceiro segmento, não mostramos essa janela nesse exemplo porque a comunicação é apenas unidirecional.

Oito segmentos são trocados entre o cliente e o servidor:

1. O primeiro segmento vai do cliente para o servidor (um segmento de SYN) para solicitar uma conexão. O cliente anuncia seu número de sequência inicial como nrSeq = 100. Quando esse segmento chega ao servidor, ele aloca um tamanho do *buffer* de 800 (uma suposição) e define sua janela de forma a cobrir o *buffer* inteiro (*rwnd* = 800). Perceba que o número do próximo *byte* que deve chegar é 101.
2. O segundo segmento vai do servidor para o cliente. É um segmento de ACK + SYN. O segmento usa nrAck = 101 para mostrar que ele espera receber *bytes* a partir do 101, e também anuncia que o cliente pode definir o tamanho do *buffer* em 800 *bytes*.
3. O terceiro segmento é o de ACK do cliente para o servidor. Perceba que o cliente tinha definido um *rwnd* de tamanho 2000, mas não usamos esse valor em nossa figura porque a comunicação se dá em apenas uma direção.
4. Após o cliente ter definido sua janela com o tamanho que foi ditado pelo servidor (800), o processo empurra 200 *bytes* de dados para o TCP. O cliente TCP numera esses *bytes* de 101 a 300. Em seguida, ele cria um segmento e o envia para o servidor. O segmento tem 101 como número do *byte* inicial e transporta 200 *bytes*. A janela do cliente é então ajustada para mostrar que 200 *bytes* de dados foram enviados, porém estão aguardando confirmação. Quando esse segmento é recebido pelo servidor, os *bytes* são armazenados e a janela de recepção fecha para mostrar que o próximo *byte* esperado é o de número 301, os *bytes* armazenados ocupam 200 *bytes* do *buffer*.
5. O quinto segmento é o retorno do servidor para o cliente. O servidor confirma a recepção dos *bytes* até e incluindo o 300 (esperando receber o *byte* 301). O segmento também carrega o tamanho da janela de recepção após a redução (600). O cliente, depois de receber esse segmento, elimina os *bytes* confirmados da sua janela e a fecha para indicar que o próximo *byte* a ser enviado é o de número 301. O tamanho da janela, no entanto, diminui para 600 *bytes*. Embora o *buffer* alocado possa armazenar 800 *bytes*, a janela não pode abrir (movendo seu limite direito mais para a direita) porque o receptor não permite.
6. O segmento 6 é enviado pelo cliente após o processo empurrar mais 300 *bytes*. O segmento define o nrSeq como sendo 301 e contém 300 *bytes*. Quando esse segmento chega ao servidor, este armazena seus *bytes*, porém precisa reduzir o tamanho da janela. Após seu processo puxar 100 *bytes* de dados, a janela se fecha à esquerda de uma quantidade de 300 *bytes*, porém se abre à direita de uma quantidade de 100 *bytes*. O resultado é que o tamanho da janela é reduzido em apenas 200 *bytes*. O tamanho da janela de recepção é agora de 400 *bytes*.
7. No segmento 7, o servidor confirma o recebimento de dados e informa que o tamanho da janela é de 400. Quando esse segmento chega ao cliente, este não tem escolha a não ser reduzir a janela novamente e definir seu tamanho para o valor de *rwnd* = 400 anunciado pelo servidor. A janela de envio se fecha a partir da esquerda de 300 *bytes* e se abre à direita de uma quantidade de 100 *bytes*.
8. O segmento 8 também é proveniente do servidor após seu processo ter puxado outros 200 *bytes*. Seu tamanho de janela aumenta. O novo valor de *rwnd* é agora 600. O segmento informa ao cliente que o servidor espera pelo *byte* 601, mas o tamanho da janela do servidor foi expandido para 600. Precisamos mencionar que o envio desse segmento depende da política imposta pela implementação. Algumas implementações podem não permitir o anúncio do *rwnd* nesse momento; nesse caso, o servidor precisa receber alguns dados antes de fazer isso. Após esse segmento chegar ao cliente, este abre sua janela em 200 *bytes*, sem fechá-la. O resultado é que o tamanho da janela aumenta para 600 *bytes*.

Encolhendo as janelas

Conforme dissemos anteriormente, a janela de recepção não pode encolher. A janela de envio, por outro lado, pode encolher se o receptor definir um valor para *rwnd* que resulte na diminuição da janela. No entanto, algumas implementações não permitem o encolhimento da janela de envio. Essa limitação não permite que o limite direito da janela de envio se mova para a esquerda. Em outras palavras, o receptor precisa manter a seguinte relação entre os valores da última e da nova confirmação e do último e do novo *rwnd* para prevenir o encolhimento da janela de envio.

$$\textbf{novo nrAck + nova } \textbf{\textit{rwnd}} \geq \textbf{último nrAck + última } \textbf{\textit{rwnd}}$$

O lado esquerdo da desigualdade representa a nova posição do limite direito em relação ao espaço dos números de sequência, enquanto o lado direito mostra a antiga posição do limite direito. Essa relação indica que o limite direito não deve se mover para a esquerda. A desigualdade é uma regra para o receptor poder verificar os seus anúncios. No entanto, perceba que ela só é válida se $E_p < E_n$; lembrando que todos os cálculos são feitos módulo 2^{32}.

Exemplo 3.18

A Figura 3.58 mostra o motivo para a regra anterior.

(*a*) A janela após o último anúncio

(*b*) A janela após o novo anúncio; a janela encolheu

Figura 3.58 Esquema para o Exemplo 3.18.

A parte (*a*) da figura mostra os valores da última confirmação e de *rwnd*. A parte (*b*) mostra a situação na qual o emissor enviou os *bytes* de 206 a 214. Os *bytes* de 206 a 209 são confirmados e eliminados. O novo anúncio, no entanto, define o novo valor de *rwnd* como 4, dado que 210 + 4 < 206 + 12. Quando a janela de envio encolhe, cria-se um problema: o *byte* 214, que já foi enviado, encontra-se fora da janela. A relação discutida anteriormente força o receptor a manter o limite direito da janela para que ele continue igual ao mostrado na parte (*a*), pois o receptor não sabe quais dos *bytes* de 210 a 217 já foram enviados. Uma maneira de evitar essa situação é deixar o receptor adiar o envio de uma mensagem de retorno até que haja uma quantidade suficiente de posições de *buffer* disponíveis em sua janela. Ou seja, o receptor deve esperar até que mais *bytes* sejam consumidos por seu processo para satisfazer a relação descrita anteriormente.

Fechamento da janela

Dissemos que encolher a janela de envio, movendo seu limite direito para a esquerda, é algo desaconselhado. No entanto, há uma exceção: o receptor pode fechar temporariamente a janela enviando um valor de *rwnd* igual a 0. Isso pode acontecer se, por algum motivo, o receptor não quiser receber qualquer dado do emissor por algum tempo. Nesse caso, o emissor não reduz de fato o tamanho da janela, mas interrompe o envio de dados até que um novo anúncio chegue. Conforme veremos mais adiante, mesmo quando a janela é fechada por ordem do receptor, o emissor sempre pode enviar um segmento com 1 *byte* de dados. Isso se chama sondagem e é utilizado para evitar uma situação de bloqueio (ver seção sobre temporizadores do TCP).

Síndrome da "janela boba"

Um problema grave pode surgir na operação de deslizamento da janela quando o aplicativo emissor produz dados lentamente, ou quando o aplicativo receptor consome dados lentamente, ou ambos. Qualquer uma dessas situações resulta no envio de dados em segmentos muito pequenos, o que reduz a eficiência do sistema. Por exemplo, se o TCP enviar segmentos contendo apenas 1 *byte* de dados, significa que um datagrama com 41 *bytes* (20 *bytes* do cabeçalho TCP e 20 *bytes* de cabeçalho IP) transfere apenas 1 *byte* de dados do usuário. Aqui, o gasto com informação de controle é de 41 para 1, o que indica que estamos utilizando a capacidade da rede de uma maneira muito ineficiente. A ineficiência é ainda pior após a contabilização das informações de controle introduzidas pelas camadas de enlace de dados e física. Esse problema é chamado **síndrome da "janela boba"** (*silly window syndrome*). Para cada lado da comunicação, descrevemos primeiramente como o problema é criado e em seguida propomos uma solução.

Síndrome criada pelo emissor

O TCP emissor pode levar à situação da síndrome da janela boba se estiver fornecendo serviços a um aplicativo que cria dados lentamente, como 1*byte* de cada vez. O aplicativo grava 1 *byte* de cada vez no *buffer* do TCP emissor. Se o TCP emissor não tiver instruções específicas de como proceder, ele pode vir a criar segmentos contendo apenas 1 *byte* de dados. O resultado é que vários segmentos de 41 *bytes* acabam trafegando pela internet.

A solução é evitar que o TCP emissor envie os dados *byte* a *byte*. Ele deve ser forçado a esperar e coletar dados para enviar em um bloco grande. Quanto tempo o TCP emissor deve esperar? Se esperar muito tempo, isso pode atrasar o processo. Se não esperar por tempo suficiente, pode acabar enviando segmentos pequenos. O pesquisador Nagle encontrou uma solução elegante para esse problema. O **Algoritmo de Nagle*** é simples:

1. O TCP emissor envia a primeira parte dos dados que recebe do aplicativo emissor, mesmo que o tamanho deles seja de apenas 1 *byte*.
2. Após enviar do primeiro segmento, o TCP emissor acumula dados no *buffer* de saida e espera até que o TCP receptor envie uma confirmação ou até que haja dados suficientes para preencher um segmento de tamanho máximo. Nesse momento, o TCP emissor pode enviar o segmento.
3. O passo 2 é repetido para o restante da transmissão. O segmento 3 é enviado imediatamente se uma confirmação do segmento 2 for recebida ou se dados suficientes tiverem sido acumulados para preencher um segmento de tamanho máximo.

A elegância do algoritmo de Nagle está na sua simplicidade e no fato de que ele leva em conta a velocidade do aplicativo que cria os dados e a velocidade da rede que os transporta. Se o aplicativo for mais rápido do que a rede, os segmentos são maiores (segmento de tamanho máximo). Se o aplicativo for mais lento do que a rede, os segmentos são menores (menores do que o tamanho máximo do segmento).

* N. de T.: O Algoritmo de Nagle é descrito na RFC 896.

Síndrome criada pelo receptor

O TCP receptor pode levar à situação da síndrome da janela boba se estiver fornecendo serviços para um aplicativo que consome dados lentamente, como um *byte* de cada vez. Considere que o aplicativo emissor crie dados em blocos de 1 *kbytes*, porém o consumo de dados no aplicativo receptor seja feito um *byte* de cada vez, e também que o *buffer* de entrada do TCP receptor seja de 4 *kbytes*. O emissor envia os primeiros 4 *kbytes* de dados. O receptor armazena os dados no seu *buffer*. Agora, o *buffer* está cheio. Ele anuncia um tamanho de janela zero, o que significa que o emissor deve parar de enviar dados. O aplicativo receptor lê o primeiro *byte* de dados do *buffer* de entrada do TCP receptor. Agora, há um *byte* de espaço livre no *buffer* de entrada. O TCP receptor anuncia um tamanho de janela de 1 *byte*, o que significa que o TCP emissor, que está esperando ansiosamente para enviar dados, recebe esse anúncio como uma boa notícia e envia um segmento carregando apenas 1 *byte* de dados. O processo continua. Um *byte* de dados é consumido e um segmento transportando 1 *byte* de dados é enviado. Novamente, temos um problema de ineficiência e o aparecimento da síndrome da janela boba.

Duas soluções foram propostas para evitar a síndrome da janela boba criada por um aplicativo que consome dados mais lentamente do que eles chegam. A **solução de Clark** é enviar uma confirmação logo que os dados chegam, mas continuar anunciando uma janela de tamanho zero até que haja espaço suficiente para acomodar um segmento de tamanho máximo ou até que pelo menos metade do *buffer* do receptor esteja vazio. A segunda solução é atrasar o envio das confirmações. Isto significa que, quando um segmento chega, ele não é confirmado imediatamente. O receptor espera até que haja uma quantidade razoável de espaço em seu *buffer* de entrada antes de confirmar os segmentos que chegaram. A confirmação tardia impede que o TCP emissor deslize a sua janela. Após o TCP emissor enviar os dados na janela, ele é interrompido. Isso elimina a síndrome.

A confirmação atrasada também tem outra vantagem: ela reduz o tráfego. O receptor não precisa confirmar cada segmento. Entretanto, há também uma desvantagem, pois uma confirmação atrasada pode fazer com que o emissor retransmita desnecessariamente os segmentos não confirmados (caso o tempo limite seja atingido).

O protocolo equilibra as vantagens e desvantagens. Ele atualmente especifica que a confirmação não deve ser atrasada por mais do que 500 ms.

3.4.8 Controle de erros

O TCP é um protocolo confiável da camada de transporte. Isto significa que um aplicativo que entrega um fluxo de dados para o TCP confia no TCP para entregar todo o fluxo para o aplicativo na outra extremidade de modo ordenado, sem erros e sem qualquer parte perdida ou duplicada.

O TCP fornece confiabilidade usando controle de erros, que inclui mecanismos para detectar e reenviar segmentos corrompidos, reenviar segmentos perdidos, armazenar segmentos fora de ordem até que os segmentos faltantes cheguem e detectar/descartar segmentos duplicados. O controle de erros no TCP é conseguido por meio do uso de três ferramentas simples: soma de verificação (*checksum*), confirmação e tempo-limite.

Soma de verificação

Cada segmento inclui um campo de soma de verificação (*checksum*) usado para verificar se há algum segmento corrompido. Se um segmento estiver corrompido, algo detectado por uma soma de verificação inválida, o segmento é descartado pelo TCP no destino e é considerado perdido. O TCP utiliza uma soma de verificação de 16 *bits* que é obrigatória em todos os segmentos. Discutimos o cálculo da soma de verificação no Capítulo 5.

Confirmações

O TCP usa confirmações para confirmar o recebimento dos segmentos de dados. Segmentos de controle que não transportam dados, mas consomem um número de sequência, também são confirmados. Segmentos de ACK nunca são confirmados.

Segmentos de ACK não consomem números de sequência e não são confirmados.

Tipo de confirmação

No passado, o TCP utilizava apenas um tipo de confirmação: a cumulativa. Atualmente, algumas implementações do TCP também usam confirmação seletiva.

Confirmação Cumulativa (ACK) O TCP foi originalmente projetado para confirmar a recepção de segmentos cumulativamente. O receptor anuncia o próximo *byte* que espera receber, ignorando todos os segmentos recebidos e armazenados fora de ordem. Isto é conhecido como ***confirmação cumulativa positiva***, ou ACK. A palavra ***positiva*** indica que nenhum retorno é fornecido para segmentos descartados, perdidos ou duplicados. O campo de ACK de 32 *bits* no cabeçalho TCP é usado para confirmações cumulativas, e seu valor é válido somente quando o *bit* do marcador ACK vale 1.

Confirmação Seletiva (SACK) Mais e mais implementações do TCP estão adicionando outro tipo de confirmação, chamada ***confirmação seletiva*** ou SACK. Um SACK não substitui um ACK, mas transmite informações adicionais ao remetente. O SACK informa quando um bloco de *bytes* está fora de ordem e também quando um bloco de *bytes* foi duplicado, ou seja, foi recebido mais de uma vez. No entanto, como não existe qualquer campo especifico disponível no cabeçalho TCP para adicionar esse tipo de informação, o pacote de SACK é implementado como uma opção no fim do cabeçalho TCP. Discutimos esse novo recurso quando abordamos as opções do TCP no *site* do livro.

Geração de confirmações

Quando um receptor gera as confirmações? Durante a evolução de TCP, várias regras foram definidas e utilizadas por diversas implementações. Apresentamos aqui as regras mais comuns. A ordem das regras não define necessariamente sua importância.

1. Quando a extremidade A envia um segmento de dados para a extremidade B, ela deve incluir (usando o mecanismo de carona, ou *piggybacking*) uma confirmação que informe o próximo número de sequência que espera receber. Essa regra diminui o número de segmentos necessários e, portanto, reduz o tráfego.
2. Quando o receptor não tem dados para enviar e recebe um segmento fora da ordem (com um número de sequência esperado), sendo que o segmento anterior já foi confirmado, o receptor atrasa o envio de um segmento de ACK até que outro segmento chegue ou até que um tempo-limite (normalmente de 500 ms) seja atingido. Em outras palavras, o receptor precisa atrasar o envio de um segmento de ACK se houver apenas um segmento pendente que esteja fora da ordem. Essa regra reduz o número de segmentos de ACK.
3. Quando um segmento chega com um número de sequência que é esperado pelo receptor e o segmento anterior recebido em ordem não foi confirmado, o receptor envia imediatamente um segmento de ACK. Em outras palavras, não deve haver mais do que dois segmentos sem confirmação que estejam em ordem em qualquer instante. Isto impede a retransmissão desnecessária de segmentos, algo que poderia criar congestionamento na rede.
4. Quando um segmento chega com um número de sequência fora de ordem que seja maior do que o esperado, o receptor imediatamente envia um ACK anunciando o número de

sequência do próximo segmento esperado. Isto leva à retransmissão rápida dos segmentos faltantes (conforme discutido mais adiante).

5. Quando um segmento que estava faltando chega ao receptor, ele envia um segmento de ACK para anunciar o próximo número de sequência esperado. Isto informa ao receptor que o segmento faltante foi recebido.
6. Se um segmento duplicado chegar ao receptor, ele descarta o segmento, porém envia imediatamente uma confirmação indicando o próximo segmento esperado, considerando a ordem correta. Isto resolve alguns problemas quando um segmento de ACK for perdido.

Retransmissão

O coração do mecanismo de controle de erros é a retransmissão de segmentos. Quando um segmento é enviado, ele é armazenado em uma fila até ser confirmado. Quando o temporizador de retransmissão expira ou quando o emissor recebe três ACKs duplicados para o primeiro segmento na fila, aquele segmento é retransmitido.

Retransmissão após o RTO

O TCP emissor gerencia um **Tempo-Limite de Retransmissão** (RTO – Retransmission Time-Out) para cada conexão. Quando o temporizador expira, ou seja, atinge o tempo-limite, o TCP reenvia o segmento do início da fila (o segmento com o menor número de sequência) e reinicia o temporizador. Perceba que, novamente, consideramos $E_p < E_n$. Veremos mais adiante que o valor do RTO é dinâmico no TCP e é atualizado com base no valor do Tempo de Ida e Volta (RTT – Round Trip Time) dos segmentos. O RTT é o tempo necessário para um segmento alcançar um destino e para que sua confirmação seja recebida.

Retransmissão após três segmentos de ACK duplicados

A regra anterior sobre a retransmissão de um segmento é suficiente se o valor do RTO não for grande. Para agilizar o serviço em toda a Internet, permitindo que os emissores retransmitam sem esperar que o tempo-limite seja atingido, a maioria das implementações de hoje segue a regra dos três ACKs duplicados e retransmite o segmento faltante imediatamente. Esse recurso é denominado **retransmissão rápida** (*fast retransmission*). Nessa versão, se três confirmações duplicadas (isto é, um ACK original mais três cópias exatamente idênticas do ACK) chegarem para um segmento, o próximo segmento é retransmitido sem esperar que o tempo-limite seja atingido. Voltamos a esse recurso mais adiante neste capítulo.

Segmentos fora de ordem

As implementações atuais do TCP não descartam segmentos fora de ordem. Elas os armazenam temporariamente e os marcam como segmentos fora da ordem até que os faltantes cheguem.

Note, no entanto, que segmentos fora de ordem nunca são entregues ao processo da camada de aplicação. O TCP garante que os dados sejam enviados ordenadamente para o processo.

> **Os dados podem chegar fora de ordem e ser temporariamente armazenados pelo TCP receptor, mas o TCP garante que nenhum dado fora de ordem seja entregue ao processo.**

MEFs para a transferência de dados no TCP

A transferência de dados no TCP é parecida com o protocolo de Repetição Seletiva, tendo uma leve similaridade com o GBN. Como o TCP aceita segmentos fora de ordem, o comportamento do TCP

pode ser comparado com o do protocolo RS, mas como as confirmações são cumulativas, ele se parece com o GBN. No entanto, se a implementação do TCP utiliza SACKS, então o TCP é mais próximo do RS.

O TCP pode ser mais bem modelado como um protocolo de Repetição Seletiva.

MEF do lado emissor

Mostraremos uma MEF simplificada para o lado emissor do protocolo TCP, que é semelhante à discutida para o protocolo RS, mas com algumas modificações especificas para o TCP. Consideramos que a comunicação é unidirecional e que os segmentos são confirmados usando segmentos de ACK. Também ignoramos, por ora, as confirmações seletivas e o controle de congestionamento. A Figura 3.59 mostra a MEF simplificada para o lado do emissor. Observe que a MEF é rudimentar; não inclui questões como a síndrome da janela boba (Algoritmo de Nagle) ou o fechamento da janela. Ela define uma comunicação unidirecional, ignorando todas as questões que afetam a comunicação bidirecional.

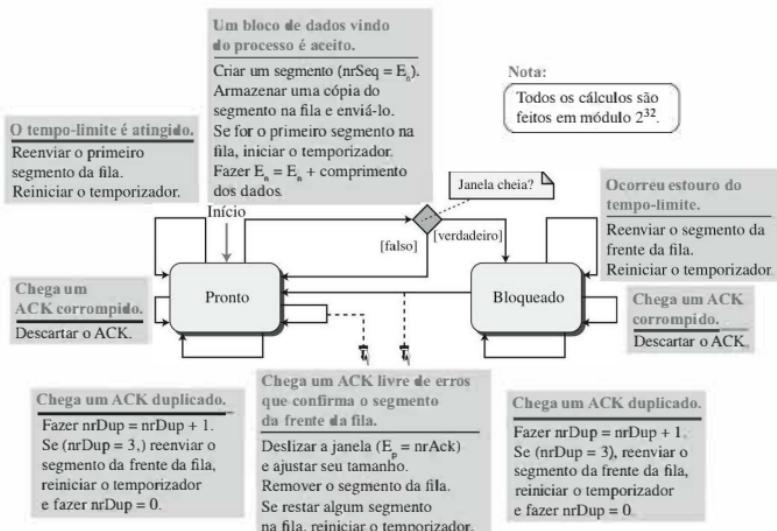



Figura 3.59 MEF simplificada para o lado emissor do TCP.

Existem algumas diferenças entre a MEF na Figura 3.59 e aquela que discutimos para o protocolo RS. Uma diferença é a retransmissão rápida (causada por três ACKs duplicados). A outra é o ajuste do tamanho da janela com base no valor de *rwnd* (ignorando o controle de congestionamento por enquanto).

MEF do lado receptor

Agora mostraremos uma MEF simplificada para o lado receptor do protocolo TCP semelhante àquela que discutimos para o protocolo RS, mas com algumas alterações específicas para o TCP. Consideramos que a comunicação é unidirecional e que os segmentos são confirmados usando segmentos de ACK. Também ignoramos a confirmação seletiva e o controle de congestionamento por ora. A Figura 3.60 mostra a MEF simplificada para o emissor. Perceba que ignoramos algumas questões, como a síndrome da janela boba (solução de Clark) e o fechamento da janela.

Novamente, existem algumas diferenças entre essa MEF e aquela que discutimos para um protocolo RS. Uma diferença é o atraso do ACK na comunicação unidirecional. A outra diferença é o envio de ACKs duplicados para permitir que o emissor possa implementar uma política de retransmissão rápida.

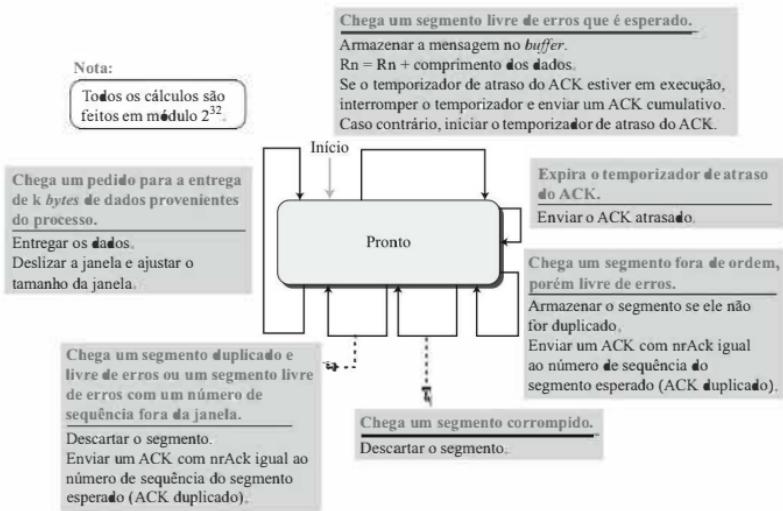



Figura 3.60 MEF simplificada para o lado receptor do TCP.

Precisamos também enfatizar que a MEF bidirecional para o receptor não é tão simples quanto aquela do RS; precisamos considerar algumas políticas, tais como o envio imediato de um ACK se o receptor possuir dados para serem enviados de volta.

Alguns cenários

Nesta seção, apresentamos alguns exemplos de situações que ocorrem durante a execução do TCP, considerando apenas questões de controle de erros. Nesses cenários, representamos um segmento por um retângulo. Se o segmento transportar dados, mostramos o intervalo de números de *bytes* carregados e o valor do campo de confirmação. Se ele carregar apenas uma confirmação, mostramos apenas o número de confirmação em uma caixa menor.

Operação normal

O primeiro cenário mostra a transferência bidirecional de dados entre dois sistemas, conforme mostrado na Figura 3.61. O TCP cliente envia um segmento; o TCP servidor envia três. A figura mostra qual regra se aplica a cada confirmação. No lado do servidor, apenas a regra 1 se aplica. Existem dados a serem enviados, de modo que o segmento exibe o próximo *byte* esperado. Quando o cliente recebe o primeiro segmento do servidor, ele não tem mais dados para enviar; precisa enviar apenas um segmento de ACK. No entanto, de acordo com a regra 2, a confirmação deve ser adiada por 500 ms para ver se chegam mais segmentos. Quando o temporizador de atraso do ACK expira, ele dispara uma confirmação. Isto acontece porque o cliente não sabe se outros segmentos estão vindo e não pode atrasar a confirmação (ACK) eternamente. Quando o próximo segmento chega, outro temporizador de atraso do ACK é inicializado. No entanto, antes que ele expire, o terceiro segmento chega. A chegada do terceiro segmento dispara outra confirmação com base na regra 3. Não mostramos o temporizador RTO porque nenhum segmento é perdido ou atrasado. Consideramos simplesmente que o temporizador RTO executa a sua tarefa.

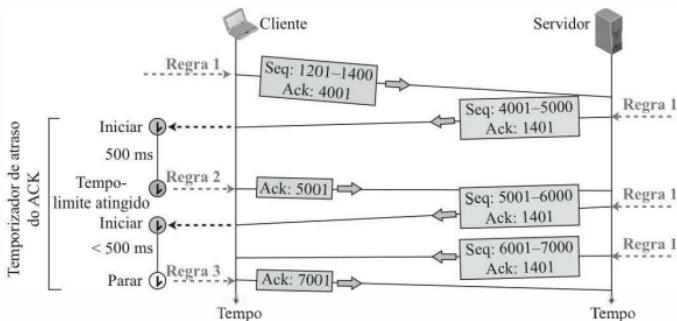



Figura 3.61 Operação normal.

Segmento perdido

Neste cenário, mostramos o que acontece quando um segmento é perdido ou corrompido. Segmentos perdidos ou corrompidos são tratados da mesma forma pelo receptor. Um segmento perdido é descartado em algum lugar da rede; um corrompido é descartado pelo próprio receptor. Ambos são considerados perdidos. A Figura 3.62 mostra uma situação na qual um segmento é perdido (provavelmente descartado por algum roteador na rede devido ao congestionamento).

Estamos considerando que a transferência de dados é unidirecional: um lado está enviando e o outro, recebendo. Em nosso cenário, o emissor envia os segmentos 1 e 2, que são reconhecidos imediatamente por um ACK (regra 3). O segmento 3, no entanto, é perdido. O receptor recebe o segmento 4, que está fora de ordem. O receptor armazena os dados do segmento no seu *buffer*, mas deixa uma lacuna no *buffer* para indicar que não há continuidade nos dados. O receptor envia imediatamente uma confirmação para o emissor que indica qual é o próximo *byte* que ele está esperando (regra 4). Perceba que o receptor armazena os *bytes* de 801 a 900, mas nunca entrega esses *bytes* para a aplicação antes que a lacuna seja preenchida.

O TCP receptor entrega apenas dados ordenados para o processo.

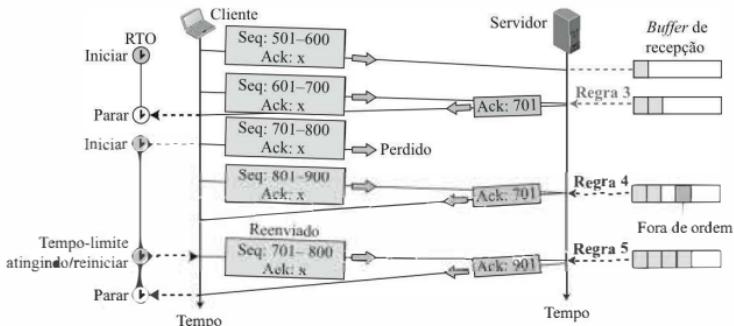



Figura 3.62 Segmento perdido.

O emissor TCP mantém um temporizador RTO durante todo o período de conexão. Quando o temporizador referente ao terceiro segmento expira, o TCP emissor reenvia o segmento 3, que dessa vez chega ao destino e é confirmado corretamente (regra 5).

Retransmissão rápida

Neste cenário, queremos mostrar o mecanismo de *retransmissão rápida* (*fast retransmission*). Nosso cenário é igual ao segundo cenário apresentado, exceto que o RTO tem um valor maior (ver Figura 3.63).

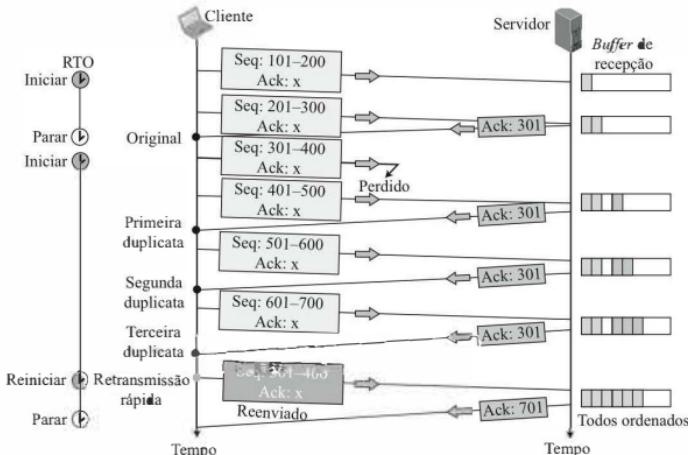



Figura 3.63 Retransmissão rápida.

Cada vez que o receptor recebe um segmento subsequente, ele responde com uma confirmação (regra 4). O emissor recebe quatro confirmações com o mesmo valor (três duplicatas). Embora o temporizador não tenha expirado, a regra para a retransmissão rápida exige que o segmento 3, o segmento que é esperado por todas essas confirmações duplicadas, seja reenviado imediatamente. Após o reenvio desse segmento, o temporizador é reiniciado.

Segmento atrasado

O quarto cenário apresenta um segmento atrasado. O TCP utiliza os serviços do IP, que é um protocolo não orientado à conexão. Cada datagrama IP encapsula um segmento TCP que pode chegar ao destino final por meio de uma rota diferente, com um atraso diferente. Por isso, alguns segmentos TCP podem ser retardados no trajeto. Os segmentos atrasados algumas vezes podem atingir o tempo-limite e serem retransmitidos. Se o segmento atrasado chegar depois de ter sido reenviado, ele é considerado um segmento duplicado e é descartado.

Segmento duplicado

Um segmento duplicado pode ser criado, por exemplo, por um TCP emissor quando um segmento é atrasado e tratado como perdido pelo receptor. Tratar o segmento duplicado é um processo simples para o TCP no destino, que espera receber um fluxo contínuo de *bytes*. Quando um segmento que chega contém um número de sequência igual ao de um segmento já recebido e armazenado, ele é descartado. Um ACK é enviado com o nrAck indicando o segmento esperado.

ACK perdido automaticamente corrigido

Este cenário mostra uma situação na qual a informação em uma confirmação perdida está contida na próxima, uma vantagem crucial da utilização de confirmações cumulativas. A Figura 3.64 mostra uma confirmação perdida enviada pelo receptor dos dados. No mecanismo de confirmação do TCP, uma confirmação perdida pode não ser sequer notada pelo TCP na origem. O TCP usa confirmações cumulativas. Podemos dizer que a próxima confirmação corrige automaticamente a perda da confirmação anterior.

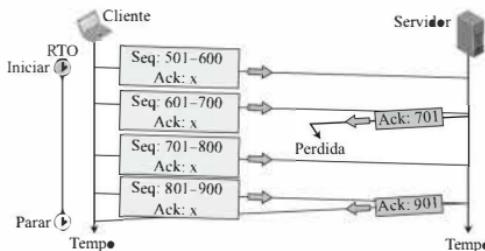



Figura 3.64 Confirmação perdida.

Correção de confirmação perdida via reenvio de segmento

A Figura 3.65 apresenta um cenário no qual a confirmação é perdida.

Se a próxima confirmação for adiada por muito tempo ou se não houver próxima (a confirmação perdida é a última enviada), a correção é acionada pelo temporizador RTO. O resultado é um segmento duplicado. Quando o receptor recebe um segmento duplicado, ele o descarta e reenvia o último ACK para informar imediatamente ao emissor que o segmento ou segmentos foram recebidos.

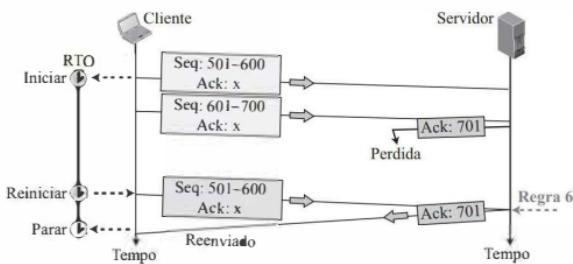



Figura 3.65 Correção de confirmação perdida via reenvio do segmento.

Observe que apenas um segmento é retransmitido, embora haja dois segmentos que não foram confirmados. Quando o emissor recebe o ACK retransmitido, ele sabe que ambos os segmentos foram recebidos corretamente porque a confirmação é cumulativa.

Criação de impasse pela confirmação perdida

Existe uma situação em que a perda de uma confirmação pode resultar em um impasse (*deadlock*) no sistema. É o caso no qual um receptor envia uma confirmação cujo *rwnd* vale 0 e solicita que o emissor feche a sua janela temporariamente. Depois de algum tempo, o receptor deseja remover essa restrição; no entanto, se ele não tiver dados para enviar, envia um segmento de ACK e remove a restrição usando um valor diferente de zero para *rwnd*. Surge um problema se essa confirmação for perdida. O emissor está esperando por uma confirmação que anuncie o valor de *rwnd* diferente de zero. O receptor pensa que o emissor recebeu tal confirmação e fica à espera de dados dele. Essa situação é chamada de **impasse** (*deadlock*); cada extremidade está esperando por uma resposta do outro lado e nada está acontecendo. Um temporizador de retransmissão não foi inicializado. Para evitar impasses, foi projetado um temporizador de persistência que estudaremos mais adiante neste capítulo.

Confirmações perdidas podem criar um impasse (*deadlock*) se não forem tratadas corretamente.

3.4.9 Controle de congestionamento no TCP

O TCP usa diferentes políticas para lidar com o congestionamento na rede. Descrevemos tais políticas nesta seção.

Janela de congestionamento

Quando discutimos o controle de fluxo no TCP, mencionamos que o tamanho da janela de envio é controlado pelo receptor usando o valor de *rwnd*, o qual é anunciado em cada segmento que trafega na direção oposta ao tráfego dos dados. A utilização dessa estratégia garante que a janela de recepção nunca seja sobrecarregada com os *bytes* recebidos (sem congestionamento no receptor). Isto, no entanto, não significa que os *buffers* intermediários, os *buffers* dos roteadores, não sejam congestionados. Um roteador pode receber dados de mais de um remetente. Independentemente do tamanho dos *buffers* de um roteador, eles podem ser sobrecarregados com dados, o que resulta na perda de alguns segmentos enviados por um emissor TCP em específico. Em outras palavras, não há congestionamento na outra extremidade da comunicação, mas pode haver congestionamento no meio do caminho. O TCP precisa se preocupar com o congestionamento no meio do caminho

porque muitos segmentos perdidos podem afetar seriamente o controle de erros. Uma maior perda de segmentos leva ao reenvio dos mesmos segmentos novamente, resultando no agravamento do congestionamento e, finalmente, no colapso da comunicação.

O TCP é um protocolo fim a fim que utiliza os serviços do IP. O congestionamento no roteador corresponde ao território do IP e deve ser tratado pelo IP. No entanto, como discutimos no Capítulo 4, o IP é um protocolo simples, sem controle de congestionamento. O TCP, em si, precisa ser responsável por tratar esse problema.

O TCP não pode ignorar o congestionamento na rede; não pode enviar segmentos de forma agressiva para a rede. O resultado dessa agressividade prejudicaria o próprio TCP, conforme dissemos anteriormente. O TCP também não pode ser muito conservador, enviando uma pequena quantidade de segmentos em cada intervalo de tempo, porque isto significa que ele não utilizará adequadamente a largura de banda disponível na rede. O TCP precisa definir políticas que acelerem a transmissão de dados quando não há congestionamento e que desacelerem a transmissão quando o congestionamento for detectado.

Para controlar o número de segmentos a serem transmitidos, o TCP usa outra variável chamada janela de congestionamento, *cwnd*, cujo tamanho é controlado pela situação do congestionamento na rede (conforme explicaremos em breve). A variável *cwnd* e a variável *rwnd* em conjunto definem o tamanho da janela de envio no TCP. A primeira está relacionada ao congestionamento no meio do trajeto (na rede); a segunda está relacionada com o congestionamento no ponto final da comunicação. O tamanho real da janela corresponde ao valor mínimo entre esses dois valores.

Tamanho atual da janela = Mínimo (*rwnd*, *cwnd*)

Detecção do congestionamento

Antes de discutir como o valor atribuído a *cwnd* deve ser definido e alterado, precisamos descrever como um TCP emissor pode detectar a possível existência de congestionamento na rede. O emissor TCP usa a ocorrência de dois eventos como sinais de congestionamento: expiração do tempo-limite e recepção de três ACKs duplicados.

O primeiro evento é quando o *tempo-limite* é atingido (*time-out*). Se um emissor TCP não recebe um ACK para um segmento ou para um grupo de segmentos antes que o tempo-limite seja atingido, ele considera que o segmento ou segmentos correspondentes foram perdidos e que a perda deve-se ao congestionamento.

Outro evento é o recebimento de três ACKs duplicados (quatro ACKs com o mesmo número de confirmação). Lembre-se que quando um receptor TCP envia um ACK duplicado, é sinal de que um segmento está atrasado, mas o envio de três ACKs duplicados é sinal da falta de um segmento, o que pode ser devido a congestionamento na rede. No entanto, o congestionamento no caso dos três ACKs duplicados pode ser menos grave do que no caso da expiração do tempo-limite. Quando um receptor envia três ACKs duplicados, significa que um segmento está faltando, porém três segmentos foram recebidos. A rede pode estar um pouco congestionada ou ter se recuperado do congestionamento.

Mostraremos mais adiante que uma versão anterior do TCP, denominada TCP Taho, tratava ambos os eventos (expiração do tempo-limite e três ACKs duplicados) da mesma forma, porém a versão mais recente do TCP, denominada TCP Reno, trata esses dois sinais de formas diferentes.

Um ponto muito interessante no congestionamento do TCP é que o emissor TCP usa apenas um tipo de retorno proveniente do outro lado da comunicação para detectar o congestionamento: os ACKs. A falta do recebimento regular e periódico de ACKs, levando à expiração do tempo-limite, é sinal de um congestionamento pesado; o recebimento de três ACKs duplicados é sinal de um congestionamento leve na rede.

Políticas de congestionamento

A política geral para o tratamento de congestionamentos no TCP é baseada em três algoritmos: partida lenta (*slow start*), prevenção de congestionamento (*congestion avoidance*) e recuperação

rápida (*fast recovery*). Discutimos primeiro cada um desses algoritmos antes de mostrarmos como o TCP muda de um mecanismo para o outro durante uma conexão.

Partida lenta: Aumento exponencial

O algoritmo de **partida lenta** (*slow start*) é baseado na ideia de que o tamanho da janela de congestionamento (*cwnd*) começa valendo 1 Tamanho Máximo de Segmento (MSS – Maximum Segment Size), porém aumenta de 1 MSS a cada confirmação que chega. Conforme discutimos anteriormente, o MSS é um valor negociado durante o estabelecimento da conexão, usando uma opção com o mesmo nome.

O nome desse algoritmo é enganoso; o algoritmo começa devagar, mas cresce exponencialmente. Para ilustrar a ideia, observemos a Figura 3.66. Consideramos que *rwnd* é muito maior do que *cwnd*, de modo que o tamanho da janela do emissor é sempre igual a *cwnd*. Também consideramos que cada segmento tem o mesmo tamanho e transporta uma quantidade de 1 MSS *bytes*. Para simplificar, podemos também ignorar a política de atraso do ACK e considerar que cada segmento é confirmado individualmente.

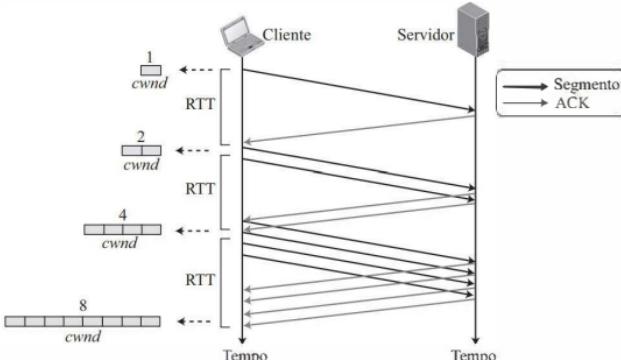



Figura 3.66 Partida lenta, aumento exponencial.

O remetente começa com *cwnd* = 1, o que quer dizer que o emissor pode enviar apenas um segmento. Após a chegada do primeiro ACK, o segmento confirmado é removido da janela, e agora há um espaço livre de segmento na janela. O tamanho da janela de congestionamento também é aumentado em 1, porque a chegada da confirmação é um bom sinal de que não há congestionamento na rede. O tamanho da janela é agora de 2. Após enviar dois segmentos e receber duas confirmações individuais para eles, o tamanho da janela de congestionamento passa a ser 4, e assim por diante. Ou seja, o tamanho da janela de congestionamento nesse algoritmo é uma função do número de ACKs recebidos e pode ser determinado como se segue.

Se um ACK chega, faz-se *cwnd* = *cwnd* + 1.

Se observarmos o tamanho de *cwnd* em termos de Tempos de Ida e Volta (RTTs – Round Trip Times), podemos ver que a taxa de crescimento é exponencial em termos de cada tempo de ida e volta, o que é uma abordagem bastante agressiva:

Início	→	$cwnd = 1 \rightarrow 2^0$
Após 1 RTT	→	$cwnd = cwnd + 1 = 1 + 1 = 2 \rightarrow 2^1$
Após 2 RTT	→	$cwnd = cwnd + 2 = 2 + 2 = 4 \rightarrow 2^2$
Após 3 RTT	→	$cwnd = cwnd + 4 = 4 + 4 = 8 \rightarrow 2^3$

O algoritmo de partida lenta não pode continuar indefinidamente. Deve haver um limiar para interromper essa fase. O emissor mantém o controle de uma variável chamada *limiar da partida lenta* (ssthresh – slow start threshold). Quando o tamanho da janela, em *bytes*, alcança esse limiar, o algoritmo de partida lenta é interrompido e a fase seguinte é iniciada.

No algoritmo de partida lenta, o tamanho da janela de congestionamento aumenta exponencialmente até atingir um limiar.

Precisamos, no entanto, mencionar que a estratégia de partida lenta é mais vagarosa caso as confirmações sejam atrasadas. Lembre-se de que, para cada ACK, o valor de *cwnd* é incrementado de apenas 1. Assim, se dois segmentos são confirmados cumulativamente, o tamanho da *cwnd* aumenta de apenas 1 e não de 2. O crescimento ainda é exponencial, mas não é uma potência de 2. Com um ACK para cada dois segmentos, ele torna-se uma potência de 1,5.

Prevenção de congestionamento: Aumento aditivo

Se continuarmos usando o algoritmo de partida lenta, o tamanho da janela de congestionamento aumentará exponencialmente. Para evitar o congestionamento antes que ele aconteça, é preciso desacelerar esse crescimento exponencial. O TCP define outro algoritmo, denominado **prevenção de congestionamento** (*congestion avoidance*) que aumenta o valor de *cwnd* aditivamente e não exponencialmente. Quando o tamanho da janela de congestionamento atinge o limiar da partida lenta (*ssthresh*) no caso em que $cwnd = i$, a fase de partida lenta é interrompida e inicia-se a fase aditiva. Nesse algoritmo, cada vez que a "janela" toda de segmentos é confirmada, o tamanho da janela de congestionamento é incrementado de um. A janela é o número de segmentos transmitidos durante um RTT. A Figura 3.67 ilustra a ideia.

O emissor começa com $cwnd = 4$. Isto significa que o emissor pode enviar apenas quatro segmentos. Depois da chegada de quatro ACKs, os segmentos confirmados são removidos da janela, e agora há espaço vazio para um segmento na janela. O tamanho da janela de congestionamento também é incrementado de 1. O tamanho da janela é agora 5. Após enviar cinco segmentos e receber as cinco confirmações para eles, o tamanho da janela de congestionamento passa a valer 6. E assim por diante. Ou seja, o tamanho da janela de congestionamento nesse algoritmo também é em função do número de ACKs recebidos e pode ser determinado como se segue:

Se um ACK chega, faz-se $cwnd = cwnd + (1/cwnd)$.

Ou seja, o tamanho da janela aumenta apenas de $1/cwnd$ do MSS (em *bytes*). Em outras palavras, todos os segmentos na janela anterior devem ser confirmados para que a janela seja aumentada em 1 MSS *bytes*.

Se observarmos o tamanho de *cwnd* em termos de RTTs, percebemos que a taxa de crescimento é linear em termos de cada RTT, o que é muito mais conservador do que a abordagem de partida lenta.

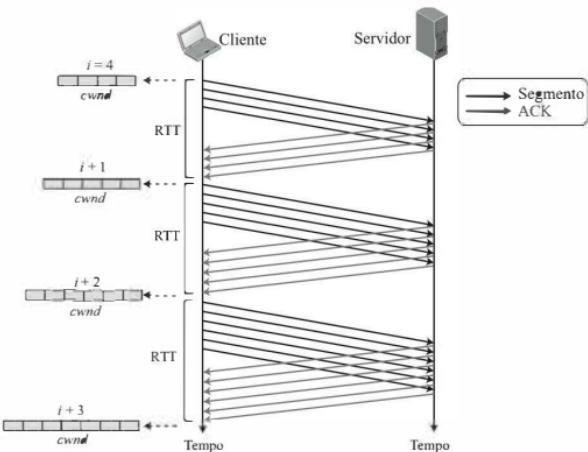



Figura 3.67 Prevenção de congestionamento, aumento aditivo.

Início	→	***cwnd*** = ***i***
Após 1 RTT	→	***cwnd*** = ***i*** + **1**
Após 2 RTT	→	***cwnd*** = ***i*** + **2**
Após 3 RTT	→	***cwnd*** = ***i*** + **3**

No algoritmo de prevenção de congestionamento, o tamanho da janela de congestionamento aumenta aditivamente até que seja detectado um congestionamento.

Recuperação rápida O algoritmo de **recuperação rápida** (*fast recovery*) é opcional no TCP. A versão antiga do TCP não o utilizava, mas as novas versões buscam utilizá-lo. Ele é iniciado quando chegam três ACKs duplicados, o que é interpretado como um congestionamento leve na rede. Assim como na prevenção de congestionamento, esse algoritmo também é um aumento aditivo, mas ele aumenta o tamanho da janela de congestionamento quando chega um ACK duplicado (a chegada de três ACKs duplicados ativa o uso desse algoritmo). Podemos dizer que

Se chegar um ACK duplicado, faz-se *cwnd* = *cwnd* + (1/*cwnd*).

Transição entre políticas

Discutimos três políticas de congestionamento no TCP. Agora, a questão é quando cada uma dessas políticas é usada e quando o TCP vai de uma política para a outra. Para responder a essas perguntas, precisamos discutir três versões do TCP: TCP Taho, TCP Reno e o TCP NewReno.

TCP Taho

O TCP inicial, conhecido como TCP Taho, usava apenas dois algoritmos diferentes em sua política de congestionamento: *partida lenta* e *prevenção de congestionamento*. Usamos a Figura 3.68 para mostrar a MEF para essa versão do TCP. No entanto, é preciso mencionar que excluímos algumas pequenas ações triviais, como incrementar e reiniciar o número de ACKs duplicados, para simplificar a MEF e deixá-la mais limpa.

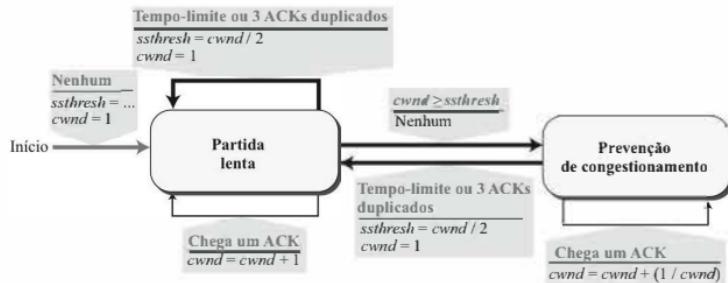



Figura 3.68 MEF para o TCP Taho.

O TCP Taho trata os dois sinais utilizados para a detecção de congestionamento, a expiração do tempo limite e a recepção de três ACKs duplicados da mesma forma. Nessa versão, quando a conexão é estabelecida, o TCP inicia o algoritmo de partida lenta e inicializa a variável *ssthresh* com um valor pré-acordado (normalmente múltiplo do MSS) e o *cwnd* com o valor de 1 MSS. Nesse estado, como dissemos anteriormente, cada vez que chega um ACK, o tamanho da janela de congestionamento é incrementado de 1. Sabemos que essa política é muito agressiva e aumenta exponencialmente o tamanho da janela, o que pode resultar em congestionamento.

Se for detectado um congestionamento (se o tempo-limite for atingido ou se forem recebidos três ACKs duplicados), o TCP interrompe imediatamente esse crescimento agressivo e reinicia um algoritmo de partida lenta, limitando o valor do limiar na metade do valor do *cwnd* atual e colocando a janela de congestionamento novamente em 1. Em outras palavras, não só o TCP recomeça o processo do início, mas também aprende a ajustar o limiar da partida lenta. Se nenhum congestionamento for detectado quando o limiar é atingido, o TCP aprende que o limite máximo de sua ambição foi atingido, não devendo continuar nessa velocidade. Ele muda para o estado de prevenção de congestionamento e se mantém nesse estado.

No estado de prevenção de congestionamento, o tamanho da janela de congestionamento é incrementado em 1 cada vez que um número de ACKs igual ao tamanho atual da janela tenha sido recebido. Por exemplo, se o tamanho da janela atualmente for de 5 MSS, cinco ACKs adicionais devem ser recebidos antes que o tamanho da janela passe para 6 MSS. Observe que não há um limite máximo para o tamanho da janela de congestionamento nesse estado; o conservador crescimento aditivo da janela de congestionamento continua até o fim da fase de transferência de dados, a menos que seja detectado um congestionamento. Nesse caso, o TCP novamente reajusta o valor de *ssthresh* para metade do valor da *cwnd* atual e vai novamente para o estado de partida lenta.

Embora nessa versão do protocolo TCP o tamanho de *ssthresh* seja continuamente ajustado em cada detecção de congestionamento, não significa necessariamente que ele se torna menor do que o valor anterior. Por exemplo, se o valor original de *ssthresh* for 8 MSS e o congestionamento for detectado quando o TCP estiver no estado de prevenção de congestionamento com o valor de *cwnd* em 20, o novo valor do *ssthresh* será então 10, o que significa que ele aumentou.

Exemplo 3.19

A Figura 3.69 mostra um exemplo de controle de congestionamento em uma rede TCP Taho. O TCP inicia a transferência de dados e ajusta a variável *ssthresh* para um valor ambicioso de 16 MSS. O TCP começa no estado de partida lenta (PL) com *cwnd* = 1. A janela de congestionamento cresce exponencialmente, mas o tempo-limite expira após o terceiro RTT (antes de o limiar ser atingido). O TCP considera que há um congestionamento na rede. Ele imediatamente reajusta o novo valor de *ssthresh* para 4 MSS (metade do valor da *cwnd* atual, que é 8) e começa novamente no estado de partida lenta (PL) com *cwnd* = 1 MSS. O congestionamento cresce exponencialmente até atingir o limiar recém-definido. O TCP vai, então, para o estado de prevenção de congestionamento (PrC) e a janela de congestionamento cresce aditivamente até atingir *cwnd* = 12 MSS. Nesse momento, três ACKs duplicados chegam, outra indicação de congestionamento na rede. O TCP reduz novamente o valor de *ssthresh* pela metade, para 6 MSS, e começa novamente no estado de partida lenta (PL). O crescimento exponencial de *cwnd* continua. Depois do RTT de número 15, o tamanho de *cwnd* é 4 MSS. Após enviar quatro segmentos e receber apenas dois ACKs, o tamanho da janela atinge *ssthresh* (6) e o TCP vai para o estado de prevenção de congestionamento. A transferência de dados continua agora no estado de prevenção de congestionamento (PrC) até que a conexão é encerrada após o RTT de número 20.

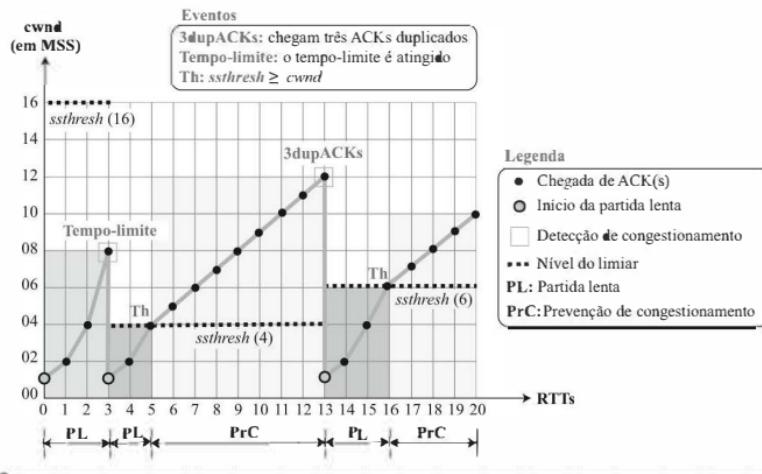



Figura 3.69 Exemplo do TCP Taho.

TCP Reno

Uma versão mais recente do TCP, denominado TCP Reno, acrescentou um novo estado à MEF do controle de congestionamento, o chamado estado de recuperação rápida. Essa versão trata os dois sinais de congestionamento, expiração do tempo-limite e chegada de três ACKs duplicados, de forma diferente. Nessa versão, se o tempo-limite for atingido, o TCP passa para o estado de partida lenta (ou começa uma nova rodada se ele já estiver nesse estado); por outro lado, se três ACKs duplicados chegarem, o TCP vai para o estado de recuperação rápida e permanece lá enquanto chegarem mais ACKs duplicados. O estado de recuperação rápida é um estado que fica de certa forma entre o estado de partida lenta e o de prevenção de congestionamento. Ele se comporta

como o estado de partida lenta, no qual o valor de *cwnd* cresce exponencialmente, porém a *cwnd* é iniciado com o valor de *ssthresh* mais 3 MSS (e não 1 MSS). Quando o TCP entra no estado de recuperação rápida, três eventos importantes podem ocorrer. Se os ACKs duplicados continuarem a chegar, o TCP permanece nesse estado, mas o valor de *cwnd* cresce exponencialmente. Se o tempo-limite for atingido, o TCP considera que há realmente um congestionamento na rede e passa para o estado de partida lenta. Se um ACK novo (não duplicado) for recebido, o TCP passa para o estado de prevenção de congestionamento, mas reduz o tamanho da *cwnd* para o valor de *ssthresh* como se a recepção dos três ACKs duplicados não tivesse ocorrido, e vai do estado de partida lenta para o de prevenção de congestionamento. A Figura 3.70 mostra a MEF simplificada para o TCP Reno. Novamente, foram removidos alguns eventos triviais para simplificar a figura e a discussão.

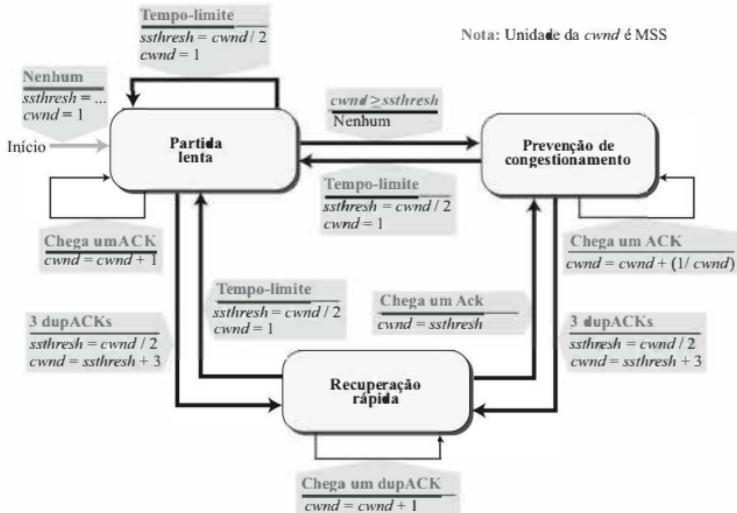



Figura 3.70 MEF para o TCP Reno.

Exemplo 3.20

A Figura 3.71 mostra a mesma situação da Figura 3.69, mas no TCP Reno. As mudanças na janela de congestionamento são as mesmas até o RTT de número 13, quando três ACKs duplicados chegam. Nesse momento, o TCP Reno reduz o *ssthresh* para 6 MSS (assim como o TCP Taho), mas coloca o valor de *cwnd* para um valor muito mais elevado (*ssthresh* + 3 = 9 MSS) e não de 1 MSS. O TCP Reno passa, então, para o estado de recuperação rápida. Consideramos que chegam mais dois ACKs duplicados até o RTT de número 15, quando o valor *cwnd* cresce exponencialmente. Nesse momento, um novo ACK (não duplicado) chega e informa o recebimento de um segmento perdido. O TCP Reno passa, então, para o estado de prevenção de congestionamento, mas primeiro reduz a janela de congestionamento para 6 MSS (o valor de *ssthresh*), como se ignorasse toda a passagem pelo estado de recuperação rápida, e volta para a situação anterior.

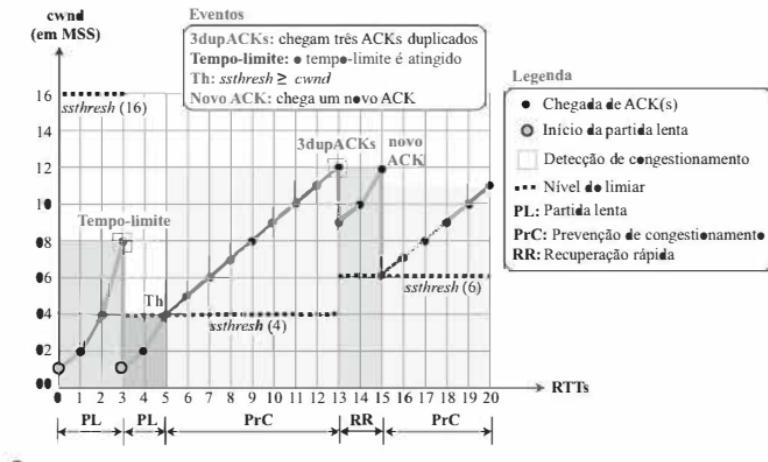



Figura 3.71 Exemplo do TCP Reno.

TCP NewReno

Uma versão mais recente do TCP, denominada TCP NewReno, fez uma otimização extra no TCP Reno. Nessa versão, o TCP verifica se mais de um segmento foi perdido na janela atual quando três ACKs duplicados chegarem. Quando o TCP recebe três ACKs duplicados, ele retransmite o segmento perdido até que um novo ACK (não duplicado) chegue. Se o novo ACK se referir à posição que se encontrava no fim da janela quando o congestionamento foi detectado, o TCP tem certeza de que apenas um segmento foi perdido. No entanto, se o número do ACK se referir a uma posição entre o segmento retransmitido e o fim da janela, é possível que o segmento indicado pelo ACK também tenha sido perdido. O TCP NewReno retransmite esse segmento para evitar a recepção de mais e mais ACKs duplicados para esse segmento.

Aumento aditivo, redução multiplicativa

Das três versões do TCP, a versão Reno é a mais comum hoje. Tem sido observado nela que, na maior parte das vezes, o congestionamento é detectado e tratado por meio do recebimento de três ACKs duplicados. Mesmo que haja alguns eventos de expiração do tempo-limite, o TCP se recupera por meio do crescimento exponencial agressivo. Em outras palavras, em uma longa conexão TCP, se ignorarmos os estados de partida lenta e o curto período de crescimento exponencial durante a recuperação rápida, a janela de congestionamento do TCP é $cwnd = cwnd + (1/cwnd)$ quando chega um ACK (prevenção de congestionamento), e $cwnd = cwnd/2$ quando um congestionamento é detectado, como se o estado de PL não existisse e a duração da RR fosse reduzida a zero. A primeira situação é conhecida como *aumento aditivo*; a segunda, como *redução multiplicativa*. Isto significa que o tamanho da janela de congestionamento, após o TCP passar pelo estado inicial de partida lenta, segue um padrão de "*dente de serra*" denominado **aumento aditivo, redução multiplicativa** (AIMD – Additive Increase, Multiplicative Decrease), conforme mostra a Figura 3.72.

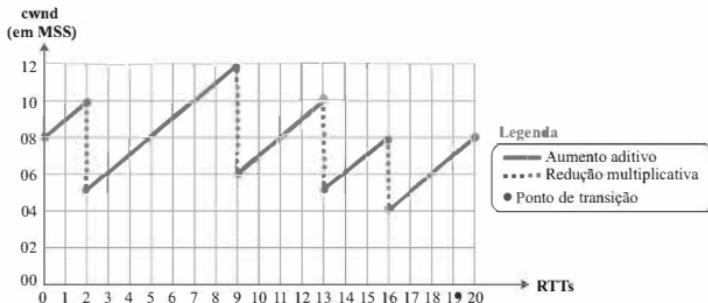



Figura 3.72 Aumento aditivo, redução multiplicativa (AIMD).

Vazão do TCP

A taxa de transferência no TCP, é baseada no comportamento da janela de congestionamento, pode ser facilmente determinada se o valor de *cwnd* for uma constante (linha horizontal) em relação ao RTT. A vazão, com essa suposição irreal, pode ser calculada como vazão = *cwnd* / RTT. Nesse cenário suposto, o TCP envia *cwnd bytes* de dados e recebe a confirmação para eles em um intervalo igual a RTT. O comportamento do TCP, conforme mostrado na Figura 3.72, não é uma linha horizontal; ele tem um formato de dentes de serra, com diversos valores mínimos e máximos. Se cada dente for exatamente do mesmo tamanho que os demais, podemos dizer que vazão = [(máximo + mínimo) / 2] / RTT. No entanto, sabemos que o valor máximo é duas vezes o valor mínimo porque em cada detecção de congestionamento, o valor de *cwnd* é ajustado para a metade do seu valor anterior. Assim, a vazão pode ser mais bem calculada como

$$\textbf{Vazão} = (0{,}75)\ W_{max}\ /\ RTT$$

onde W_{max} é a média dos tamanhos das janelas quando o congestionamento ocorre.

Exemplo 3.21

Se MSS = 10 KB (*quilobytes*) e RTT = 100 ms na Figura 3.72, podemos calcular a vazão como mostramos a seguir.

W_{max} = (10 + 12 + 10 + 8 + 8) / 5 = 9,6 MSS
Vazão = (0,75 W_{max} / RTT) = 0,75 × 960 kbps / 100 ms = 7,2 Mbps

3.4.10 Temporizadores do TCP

Para realizar suas operações sem percalços, a maioria das implementações do TCP usa pelo menos quatro temporizadores: retransmissão, persistência, manutenção de sessão (*keepalive*[*]) e TEMPO-DE-ESPERA.

[*] N. de T.: O nome *keepalive* pode ser traduzido como "manter vivo". No contexto de protocolos de rede, o termo normalmente se refere a mensagens específicas para manter conexões abertas, para que elas não sejam encerradas por falta de atividade.

Temporizador de retransmissão

Para retransmitir segmentos perdidos, o TCP emprega um temporizador de retransmissão (por toda a duração da conexão) que trata a expiração do tempo-limite de retransmissão (RTO – Retransmission Time-Out), o tempo de espera pela confirmação de um segmento. Podemos definir as seguintes regras para o temporizador de retransmissão:

1. Quando o TCP envia o segmento da frente da fila de envio, ele inicia o temporizador.
2. Quando o temporizador expira, o TCP reenvia o primeiro segmento da frente da fila e reinicia o temporizador.
3. Quando um segmento ou segmentos são confirmados cumulativamente, o segmento ou segmentos são removidos da fila.
4. Se a fila estiver vazia, o TCP interrompe o temporizador, caso contrário, o TCP reinicia o temporizador.

Tempo de Ida e Volta

Para calcular o tempo-limite de retransmissão (RTO), primeiro precisamos calcular o valor do **Tempo de Ida e Volta** (RTT – Round-Trip Time). No entanto, o cálculo do RTT no TCP é um processo complexo que explicamos passo a passo com alguns exemplos.

- **RTT medido.** Precisamos determinar quanto tempo leva para enviar um segmento e receber uma confirmação para ele. Esse tempo é o RTT medido. Precisamos lembrar que os segmentos e suas confirmações não têm uma relação um para um; diversos segmentos podem ser confirmados juntos. O tempo de ida e volta medido para um segmento é o necessário para o segmento chegar ao destino e ser confirmado, embora a confirmação possa incluir outros segmentos. Observe que, no TCP, apenas uma medição do RTT pode estar em andamento de cada vez. Isto significa que se uma medição de RTT for iniciada, nenhuma outra medição pode começar até que o valor desse RTT tenha sido determinado. Usamos a notação RTT_M para representar um RTT medido.

No TCP, pode haver somente uma medição de RTT em andamento a qualquer momento.

- **RTT suavizado.** O RTT medido, RTT_M, provavelmente muda a cada ida e volta. A flutuação é tão elevada na Internet de hoje que uma única medição por si só não pode ser usada para fins de retransmissão por expiração do tempo-limite. A maioria das implementações usa um RTT suavizado, denotado RTT_S, que é uma média ponderada do RTT_M e do RTT_S anterior, conforme mostrado a seguir:

Inicialmente	→	**Sem valor**
Após a primeira medição	→	$RTT_S = RTT_M$
Após cada medição	→	$RTT_S = (1 - \alpha)\ RTT_S + \alpha \times RTT_M$

O valor de α depende da implementação, mas ele é normalmente fixado em 1/8. Em outras palavras, o novo RTT_S é calculado a partir de 7/8 do RTT_S antigo e 1/8 do RTT_M corrente.

- **Desvio do RTT.** A maioria das implementações não usa apenas o RTT_S; elas também calculam o desvio do RTT, denotado RTT_D, com base nos valores de RTT_S e RTT_M. Elas utilizam as seguintes fórmulas (o valor do β também é dependente da implementação, mas geralmente é fixado em 1/4):

Inicialmente	→	**Sem valor**
Após a primeira medição	→	$\mathbf{RTT_D = RTT_M /2}$
Após cada medição	→	$\mathbf{RTT_D = (1 - \beta)\ RTT_D + \beta \times \mid RTT_S\ RTT_M}$

Tempo Limite de Retransmissão (RTO) O valor do Tempo Limite de Retransmissão (RTO – Retransmission Time-Out) baseia-se no RTT suavizado (RTT_S) e no seu desvio (RTT_D). A maioria das implementações usa a seguinte fórmula para calcular o RTO:

Original	→	**Valor inicial**
Após qualquer medição	→	$\mathbf{RTO + RTT_S + 4 \times RTT_D}$

Em outras palavras, tome o valor médio do RTT_S que está sendo executado e adicione quatro vezes o valor do RTT_D (normalmente um valor pequeno).

Exemplo 3.22

Usaremos um exemplo hipotético. A Figura 3.73 mostra parte de uma conexão, mostrando o estabelecimento dela e parte das fases de transferência de dados.

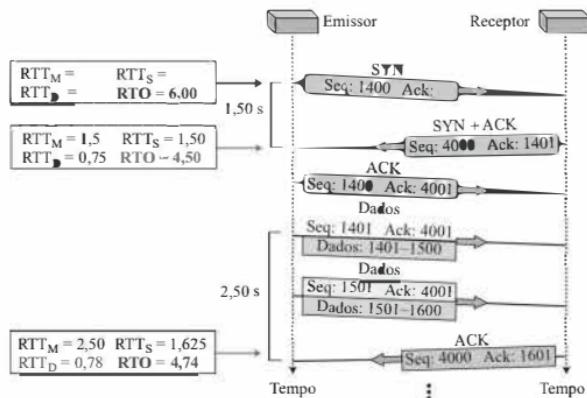



Figura 3.73 Esquema para o Exemplo 3.22.

1. Quando o segmento de SYN é enviado, não há valores definidos para RTT_M, RTT_S ou RTT_D. O valor do RTO é inicializado em 6,00 segundos. A seguir, são mostrados os valores dessas variáveis nesse momento:

RTO = 6

2. Quando chega o segmento de SYN+ACK, o RTT_M é medido e é igual a 1,5 segundos. A seguir, são mostrados os valores dessas variáveis:

$$RTT_M = 1{,}5$$
$$RTT_S = 1{,}5$$
$$RTT_D = (1{,}5)/2 = 0{,}75$$
$$RTO = 1{,}5 + 4 \times 0{,}75 = 4{,}5$$

3. Quando o primeiro segmento de dados é enviado, começa uma nova medição do RTT. Perceba que o emissor não inicia uma medição do RTT quando ele envia o segmento de ACK, pois esse ACK não consome um número de sequência e não há qualquer limite de tempo. Nenhuma medição do RTT é iniciada para o segundo segmento de dados porque uma medição já está em andamento. A chegada do último segmento de ACK é usada para calcular o valor do próximo RTT_M. Embora o último segmento de ACK confirme ambos os segmentos de dados (de forma cumulativa), a sua chegada finaliza o cálculo do valor de RTT_M para o primeiro segmento. Os valores dessas variáveis são agora aqueles mostrados a seguir.

$$RTT_M = 2{,}5$$
$$RTT_S = (7/8) \times (1{,}5) + (1/8) \times (2{,}5) = 1{,}625$$
$$RTT_D = (3/4) \times (0{,}75)(1/4) \times |\,1{,}625 - 2{,}5| = 0{,}78$$
$$RTO = 1{,}625 + 4 \times (0{,}78) = 4{,}74$$

Algoritmo de Karn

Considere que um segmento não seja confirmado durante o período do tempo-limite de retransmissão e, portanto, seja retransmitido. Quando o TCP emissor recebe uma confirmação para esse segmento, ele não sabe se o aviso refere-se ao segmento original ou ao segmento retransmitido. O novo valor do RTT baseia-se no instante de saída do segmento. No entanto, se o segmento original foi perdido e a confirmação refere-se ao segmento retransmitido, o valor do RTT corrente deve ser calculado a partir do momento em que o segmento foi retransmitido. Essa ambiguidade foi resolvida por Karn. O **algoritmo de Karn** é simples. Ele determina que não se deve considerar o tempo de ida e volta de um segmento retransmitido no cálculo do RTT. Além disso, também especifica que não se deve atualizar o valor do RTT até que se tenha enviado um segmento e recebido uma confirmação sem a necessidade de retransmissão.

O TCP não considera o RTT de segmentos retransmitidos no cálculo de um novo RTO.

Recuo exponencial

Qual é o valor de RTO se ocorrer uma retransmissão? A maioria das implementações do TCP usa uma estratégia de recuo exponencial (*exponential backoff*). O valor do RTO é dobrado para cada retransmissão. Portanto, se o segmento for retransmitido uma vez, o valor será duas vezes o RTO original. Se for duas vezes, o valor será quatro vezes o do RTO original, e assim por diante.

Exemplo 3.23

A Figura 3.74 é uma continuação do exemplo anterior. Há uma retransmissão e o algoritmo de Karn é aplicado. O primeiro segmento apresentado na figura é enviado, mas é perdido. O temporizador RTO expira após 4,74 segundos. O segmento é retransmitido e o temporizador é ajustado para 9,48 segundos, o dobro do valor anterior do RTO. Dessa vez, um ACK é recebido antes da expiração do tempo-limite. Esperamos até que um novo segmento seja enviado e que seu ACK seja recebido antes de recalcular o RTO (algoritmo de Karn).

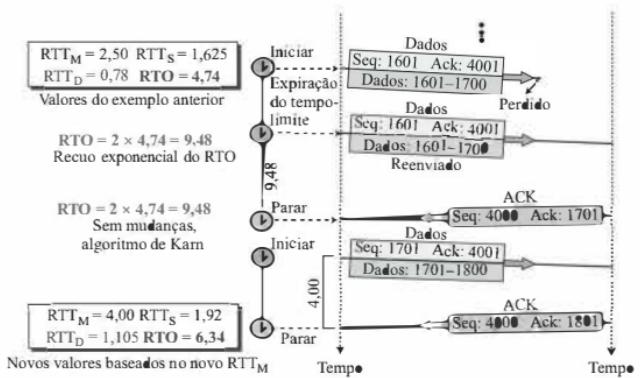



Figura 3.74 Esquema para o Exemplo 3.23.

Temporizador de persistência

Para lidar com um anúncio de uma janela com tamanho zero, o TCP precisa de outro temporizador. Se o TCP receptor anuncia um tamanho de janela de zero, o TCP emissor interrompe a transmissão de segmentos até que o TCP receptor envie um segmento de ACK anunciando um tamanho de janela diferente de zero. Esse segmento de ACK pode ser perdido. Lembre-se de que os segmentos de ACK não são confirmados nem retransmitidos pelo TCP. Se essa confirmação for perdida, o TCP receptor supõe que fez o seu trabalho e espera que o TCP emissor envie mais segmentos. Não há um temporizador de retransmissão para segmentos contendo apenas uma confirmação (ACK). No outro lado, o TCP emissor não recebeu uma confirmação e aguarda que o TCP receptor envie uma confirmação anunciando o tamanho da janela. Ambos os TCPs podem continuar esperando um pelo outro indefinidamente (tem-se um impasse).

Para corrigir esse impasse, o TCP usa um **temporizador de persistência** para cada conexão. Quando o TCP emissor recebe uma confirmação com um tamanho de janela zero, ele inicia um temporizador de persistência. Quando o temporizador de persistência expira, o TCP emissor envia um segmento especial chamado *sonda*, que contém apenas um *byte* de dados novos. Ele possui um número de sequência, que nunca será confirmado; ele é até mesmo ignorado no cálculo do número de sequência para o restante dos dados. A sonda faz com que o TCP receptor reenvie a confirmação.

O valor do temporizador de persistência é ajustado para o valor do tempo de retransmissão. No entanto, se uma resposta não for recebida do receptor, outro segmento de sonda é enviado e o valor do temporizador de persistência é dobrado e reiniciado. O emissor continua a enviar os segmentos de sonda, dobrando e reiniciando o valor do temporizador de persistência até que seu valor atinja um limite (normalmente 60 s). Depois disso, o emissor envia um segmento de sonda a cada 60 segundos até que a janela seja reaberta.

Temporizador de manutenção de sessão

Um **temporizador de manutenção de sessão** (*keepalive*) é usado em algumas implementações para evitar uma longa inatividade entre dois TCP. Considere que um cliente estabeleça uma conexão TCP com um servidor, transfira alguns dados e então fique em silêncio. Talvez o cliente tenha parado de funcionar. Nesse caso, a conexão permanece aberta para sempre.

Para remediar a situação, a maioria das implementações adota um servidor com um temporizador de manutenção de sessão. Cada vez que o servidor recebe informação de um cliente, ele reinicia o temporizador. O tempo-limite é normalmente de duas horas. Se o servidor não receber pacotes do cliente depois de duas horas, ele envia um segmento de sonda. Se não houver resposta depois de 10 sondas, cada uma das quais é espaçada de 75 segundos, ele considera que o cliente está desligado e finaliza a conexão.

Temporizador de TEMPO-DE-ESPERA

O temporizador de TEMPO-DE-ESPERA (2MSL) é usado durante a finalização de uma conexão. O Tempo de Vida Máximo de Segmento (MSL – Maximum Segment Life) é o tempo que qualquer segmento pode existir em uma rede até ser descartado. Cada implementação precisa escolher um valor para o MSL. Os valores comuns são 30 segundos, um minuto ou até mesmo dois minutos. O temporizador 2MSL é usado quando o TCP executa um fechamento ativo e envia o ACK final. A conexão deve ser mantida durante o período de tempo de 2 MSL para permitir que o TCP reenvie o ACK final caso esse ACK seja perdido. Isto requer que o temporizador RTO na outra extremidade expire e novos segmentos de FIN e de ACK sejam reenviados.

3.4.11 Opções

O cabeçalho TCP pode ter até 40 *bytes* de informações opcionais. As opções contêm informações adicionais para o destino ou se referem a outras opções, que foram incluídas no *site* do livro para referência futura.

As opções do TCP são discutidas na página Web do livro.

3.5 MATERIAL DO FINAL DO CAPÍTULO

3.5.1 Leitura adicional

Para mais detalhes sobre os assuntos discutidos neste capítulo, recomendamos os seguintes livros e RFCs. Os itens entre colchetes [...] referem-se à lista de referências no fim do texto.

Livros

Diversos livros fornecem informações sobre protocolos da camada de transporte. Em particular, recomendamos a leitura de [Com 06], [Pet & Dav 03], [Gar & Wid 04], [Far 04], [Tan 03] e [Sta 04].

RFCs

A principal RFC relacionada ao UDP é a RFC 768. Diversas RFCs discutem o protocolo TCP, incluindo RFC 793, RFC 813, RFC 879, RFC 889, RFC 896, RFC 1122, RFC 1975, RFC 1987, RFC 1988, RFC 1993, RFC 2018, RFC 2581, RFC 3168 e RFC 3782.

3.5.2 Termos-chave

- algoritmo de Karn
- algoritmo de Nagle
- ataque de inundação de SYN
- ataque de negação de serviço

- Aumento Aditivo, Redução Multiplicativa (AIMD – Additive Increase, Multiplicative Decrease)
- comunicação processo a processo
- congestionamento
- controle de congestionamento
- *cookie*
- datagrama de usuário
- demultiplexação
- encadeamento (*pipelining*)
- endereço de *socket*
- impasse (*deadlock*)
- janela deslizante
- Máquina de Estados Finitos (MEF)
- mecanismo de carona (*piggybacking*)
- multiplexação
- número de confirmação
- número de porta bem conhecido
- número de porta efêmera
- número de sequência
- número de sequência inicial
- partida lenta
- produto largura de banda-atraso
- Protocolo de Controle de Transmissão (TCP – Transmission Control Protocol)
- Protocolo de Datagramas de Usuário (UDP – User Datagram Protocol)
- protocolo de Repetição Seletiva (RS)
- protocolo *Go-Back-N* (Volte Atrás N)
- protocolo *Stop-and-Wait* (Pare e Espere)
- retransmissão rápida
- segmento
- síndrome da janela boba (*silly window syndrome*)
- solução de Clark
- Tempo de Ida e Volta (RTT – Round Trip Time)
- Tempo-Limite de Retransmissão (RTO – Retransmission Time-Out)
- temporizador de manutenção de sessão (*keepalive*)
- temporizador de persistência
- *three-way handshaking* (apresentação em três vias)

3.5.3 Resumo

O principal dever de um protocolo de camada de transporte é proporcionar comunicação processo a processo. Para definir os processos, precisamos de números de porta. O programa-cliente define a si mesmo usando um número de porta efêmera. O servidor é definido com um número de porta bem conhecido. Para enviar uma mensagem de um processo para outro, o protocolo da camada de transporte encapsula e desencapsula mensagens. A camada de transporte na origem realiza multiplexação; a camada de transporte no destino executa demultiplexação. O controle de fluxo equilibra a troca de dados entre um produtor e um consumidor. Um protocolo da camada de transporte pode fornecer dois tipos de serviços: não orientados e orientados à conexão. Em um serviço não orientado à conexão, o emissor envia pacotes para o receptor, sem qualquer estabelecimento de conexão. Em um serviço orientado à conexão, o cliente e o servidor precisam primeiro estabelecer uma conexão entre eles.

Discutimos vários protocolos comuns da camada de transporte neste capítulo. O protocolo *Stop-and-Wait* (Pare e Espere) fornece tanto controle de fluxo como controle de erros, mas é ineficiente. O protocolo *Go-Back-N* (Volte Atrás *N*) é uma versão mais eficiente do protocolo *Stop-and-Wait* e tira proveito de encadeamento (*pipelining*). O protocolo de Repetição Seletiva, uma modificação do protocolo *Go-Back-N*, é o mais adequado para lidar com a perda de pacotes. Todos esses protocolos podem ser implementados bidirecionalmente usando mecanismo de carona (*piggybacking*).

O Protocolo de Datagramas de Usuário (UDP – User Datagram Protocol) é um protocolo de transporte que cria uma comunicação processo a processo. O UDP é um protocolo não confiável (a maioria das vezes) e não orientado à conexão que introduz pouca carga adicional de controle e oferece uma entrega rápida. O pacote do UDP é chamado datagrama de usuário.

O Protocolo de Controle de Transmissão (TCP – Transmission Control Protocol) é outro protocolo da camada de transporte pertencente à pilha de protocolos TCP/IP. O TCP fornece um serviço processo a processo, *full-duplex* e orientado à conexão. A unidade de transferência de dados entre dois dispositivos usando o TCP é chamada segmento. Uma conexão TCP consiste em três fases: estabelecimento da conexão, transferência de dados e finalização da conexão. O TCP é normalmente implementado como uma máquina de estados finitos (MEF).

3.6 ATIVIDADES PRÁTICAS

3.6.1 Testes

Diversos testes interativos relativos a este capítulo podem ser encontrados no *site* www.mhhe.com/forouzan. Recomendamos que o estudante realize os testes para verificar seu entendimento do material apresentado antes de continuar com as atividades práticas.

Questões

3-1 Considere que temos um conjunto de computadores dedicados em um sistema, cada um projetado para executar uma única tarefa. Ainda precisamos de comunicação *host* a *host* e processo a processo, além de dois níveis de endereçamento?

3-2 Os sistemas operacionais atribuem um número de processo para cada aplicativo em execução. Você pode explicar por que esses números de processo não podem ser usados em vez dos números de porta?

3-3 Considere que você precise escrever e testar um aplicativo cliente-servidor para duas estações que tem em casa.

a. Qual é o intervalo de números de porta que você escolheria para o programa-cliente?

b. Qual é o intervalo de números de porta que você escolheria para o programa-servidor?

c. Os dois números de porta podem ser iguais?

3-4 Considere que uma nova organização precisa criar um novo processo-servidor e permitir que seus clientes acessem a página Web da organização usando esse processo. Como deve ser selecionado o número da porta para o processo-servidor?

3-5 Em uma rede, o tamanho da janela de recepção é de um pacote. Qual dos seguintes protocolos está sendo usado?

a. *Stop-and-Wait*

b. *Go-Back-N*

c. Repetição Seletiva

3-6 Em uma rede, o tamanho da janela de envio é de 20 pacotes. Qual dos seguintes protocolos está sendo usado?

a. *Stop-and-Wait*

b. *Go-Back-N*

c. Repetição Seletiva

3-7 Em uma rede com valor fixo para $m > 1$, temos a opção de usar o protocolo *Go-Back-N* ou Repetição Seletiva. Descreva as vantagens e desvantagens de cada um. Quais outros critérios de redes devem ser considerados para selecionar algum desses protocolos?

3-8 Como o campo que armazena o número de sequência de um pacote tem tamanho limitado, o número de sequência em um protocolo precisa ser cíclico, o que significa que dois pacotes podem ter o mesmo número de sequência. Em um protocolo que utiliza m *bits* para o campo de número de sequência, se um pacote tiver o número de sequência x, quantos pacotes precisam ser enviados para encontrarmos um pacote com o mesmo número de sequência x, considerando que cada pacote recebe um número de sequência?

3-9 A situação de reinício cíclico que discutimos na questão anterior pode criar problemas em uma rede?

3-10 Por que alguns pacotes da camada de transporte podem ser recebidos fora de ordem na Internet?

3-11 Por que alguns pacotes da camada de transporte podem ser perdidos na Internet?

3-12 Por que alguns pacotes da camada de transporte podem ser duplicados na Internet?

3-13 No protocolo *Go-Back-N*, o tamanho da janela de envio pode ser $2m - 1$, enquanto o tamanho da janela de recepção é apenas 1. Como pode ser realizado o controle de fluxo quando há uma grande diferença entre o tamanho das janelas de envio e de recepção?

3-14 No protocolo de Repetição Seletiva, o tamanho das janelas de envio e de recepção é o mesmo. Isso significa que supostamente não há pacotes em trânsito?

3-15 Alguns aplicativos podem usar os serviços de dois protocolos da camada de transporte (UDP ou TCP). Quando um pacote chega ao destino, como o computador descobre qual camada de transporte está envolvida?

3-16 Um cliente residindo em um *host* com endereço IP 122.45.12.7 envia uma mensagem para um servidor residindo em um *host* com o endereço IP 200.112.45.90. Se a porta bem conhecida for 161 e a porta efêmera for 51000, quais são os pares de endereços de *socket* usados na comunicação?

3-17 O UDP é um protocolo orientado a mensagens. O TCP é um protocolo orientado a *bytes*. Se um aplicativo precisar proteger os limites de sua mensagem, qual protocolo deve ser utilizado, o UDP ou o TCP?

3-18 No Capítulo 2, dissemos que podemos ter diferentes Interfaces de Programação de Aplicativos (APIs – Application Programming Interfaces), tais como a interface *socket*, TLI e STREAM, que podem ser usadas para fornecer comunicação cliente-servidor. Isso significa que devemos ter diferentes protocolos UDP ou TCP que suportam cada uma dessas APIs? Explique.

3-19 Considere que uma internet privada, que utiliza comunicação ponto a ponto entre os *hosts* e não necessita de roteamento, eliminou totalmente o uso da camada de rede. Essa internet ainda pode se beneficiar dos serviços do UDP ou do TCP? Em outras palavras, os datagramas de usuário ou segmentos podem ser encapsulados nos quadros Ethernet?

3-20 Considere que uma internet privada utiliza uma pilha de protocolos totalmente diferente da pilha de protocolos TCP/IP. Essa internet ainda pode usar os serviços do UDP ou do TCP como uma forma de transmitir mensagens fim a fim?

3-21 Por que precisamos de quatro (ou às vezes três) segmentos para finalizar uma conexão TCP?

3-22 No TCP, alguns tipos de segmento só podem ser usados para controle; eles não podem ser usados para transportar dados ao mesmo tempo. Descreva alguns desses segmentos.

3-23 No TCP, como podemos definir o número de sequência de um segmento (em cada direção da comunicação)? Considere dois casos: o primeiro segmento e os outros segmentos.

3-24 No TCP, temos dois segmentos consecutivos. Suponha que o número de sequência do primeiro segmento seja 101. Qual é o número de sequência do próximo segmento em cada um dos seguintes casos?

a. O primeiro segmento não consome qualquer número de sequência.

b. O primeiro segmento consome 10 números de sequência.

3-25 No TCP, quantos números de sequência são consumidos por cada um dos seguintes segmentos?

a. SYN **b.** ACK

c. SYN + ACK **d.** Dados

3-26 Explique por que no TCP os segmentos de SYN, SYN + ACK e FIN consomem um número de sequência, porém um segmento de ACK sem dados não?

3-27 Observando o cabeçalho do TCP (Figura 3.44), vemos que o número de sequência tem 32 *bits* de comprimento, enquanto o tamanho da janela tem apenas 16 *bits* de comprimento. Isso significa que o TCP é mais próximo, nesse sentido, do protocolo *Go-Back-N* ou do protocolo de Repetição Seletiva?

3-28 O tamanho máximo da janela do TCP foi originalmente projetado para ser de 64 KB (o que significa $64 \times 1024 = 65.536$, ou mais precisamente 65.535). Você consegue imaginar algum motivo para isso?

3-29 Qual é o tamanho máximo do cabeçalho TCP? Qual é o tamanho mínimo do cabeçalho TCP?

3-30 No TCP, o segmento de SYN abre uma conexão em apenas uma direção ou em ambas as direções?

3-31 No TCP, o segmento de FIN fecha uma conexão em apenas uma direção ou em ambas as direções?

3-32 No TCP, que tipo de marcador pode finalizar totalmente a comunicação nos dois sentidos?

3-33 Em sua maioria, os marcadores no TCP podem ser usados conjuntamente em um segmento. Dê um exemplo de dois marcadores que não podem ser utilizados simultaneamente por serem ambíguos.

3-34 Considere que um cliente envia um segmento de SYN a um servidor. Quando o servidor verifica o número de porta bem conhecido, ele descobre que nenhum processo referente ao número de porta especificado no segmento está em execução. O que é esperado que o servidor faça nesse caso?

3-35 Explique como o TCP, que utiliza os serviços prestados pelo não confiável protocolo IP, é capaz de fornecer comunicação confiável.

3-36 Dissemos que o TCP fornece um serviço orientado à conexão entre dois programas aplicativos. Uma conexão, nesse caso, requer um identificador de conexão que diferencie uma conexão de outras. Qual você acha que é o identificador único de conexão nesse caso?

3-37 Considere que Alice usa o seu navegador para abrir duas conexões com o servidor HTTP em execução no servidor de Bob. Como essas duas conexões podem ser distinguidas pelo TCP?

3-38 Usamos os termos *abertura passiva* e *abertura ativa* para discutir uma comunicação orientada à conexão usando TCP. Considere que haja uma conversa telefônica entre Alice e Bob. Como uma conversa telefônica é um exemplo de comunicação orientada à conexão, Alice chama Bob e eles conversam por telefone. Quem está fazendo a *abertura passiva* da conexão nesse caso? Quem está fazendo a *abertura ativa*?

3-39 No TCP, a janela do emissor pode ser menor, maior ou do mesmo tamanho que a janela do receptor?

3-40 Liste algumas tarefas que podem ser feitas por um segmento ou por uma combinação de segmentos TCP.

3-41 Em um segmento TCP, o que um número de sequência identifica?

3-42 Em um segmento TCP, o que um número de confirmação identifica?

3-43 O uso da soma de verificação (*checksum*) para controle de erros é opcional ou obrigatório no

a. UDP? **b.** TCP?

3-44 Considere que um servidor TCP espera receber o *byte* de número 2.001, mas recebe um segmento com o número de sequência 2.200. Qual é a reação do servidor TCP nesse caso? Você consegue justificar tal reação?

3-45 Considere que um servidor TCP espera receber o *byte* de número 2.001, mas recebe um segmento com o número de sequência 1.201. Qual é a reação do servidor TCP nesse caso? Você consegue justificar tal reação?

3-46 Considere que em um servidor TCP estão faltando os *bytes* de 2.001 a 3.000. O servidor recebe um segmento com o número de sequência 2.001, que carrega 400 *bytes*. Qual é a reação do servidor TCP nesse caso? Você consegue justificar tal reação?

3-47 Considere que um servidor TCP espera receber o *byte* de número 2.401. Ele recebe um segmento com o número de sequência 2.401, que carrega 500 *bytes*. Se o servidor não tiver dados para enviar nesse momento e não tiver confirmado o segmento anterior, qual será a reação do servidor TCP? Você consegue justificar tal reação?

3-48 Considere que um cliente TCP espera receber o *byte* de número 3.001. Ele recebe um segmento com o número de sequência 3.001, que carrega 400 *bytes*. Se o cliente não tiver dados para enviar nesse momento e tiver confirmado o segmento anterior, qual é a reação dele? Você consegue justificar tal reação?

3-49 Considere que um servidor TCP espera receber o *byte* de número 6.001. Ele recebe um segmento com o número de sequência 6.001, que carrega 2.000 *bytes*. Se o servidor tiver os *bytes* de 4.001 a 5.000 para enviar, qual deve ser a reação do servidor TCP? Você consegue justificar tal reação?

3-50 A primeira regra na geração de ACKs do TCP não é mostrada nas Figuras 3.59 e 3.60. Explique a razão para isso.

3-51 Quais das seis regras que descrevemos para a geração de ACKs podem ser aplicadas no caso em que um servidor recebe um segmento de SYN vindo de um cliente?

3-52 Quais das seis regras que descrevemos para a geração de ACKs podem ser aplicadas no caso em que um cliente recebe um segmento de SYN + ACK vindo de um servidor?

3-53 Quais das seis regras que descrevemos para a geração de ACKs podem ser aplicadas no caso em que um servidor recebe um segmento de FIN vindo da outra extremidade?

Problemas

3-1 Compare o intervalo de endereços de 16 *bits*, de 0 a 65.535, com o intervalo de endereços IP de 32 *bits*, de 0 a 4.294.967.295 (discutido no Capítulo 4). Por que precisamos de uma faixa tão grande de endereços IP, mas de apenas um intervalo relativamente pequeno de números de porta?

3-2 Explique por que a ICANN dividiu os números de porta em três grupos: bem conhecidos, registrados e dinâmicos.

3-3 Um remetente envia uma série de pacotes para o mesmo destino usando números de sequência de 5 *bits*. Se os números de sequência começam em 0, qual é o número de sequência do centésimo pacote?

3-4 Em cada um dos seguintes protocolos, quantos pacotes podem ter números de sequência independentes antes de ocorrer o reinício da numeração cíclica (ver problemas anteriores)?

a. *Stop-and-Wait*

b. *Go-Back-N* com $m = 8$

c. Repetição Seletiva com $m = 8$

3-5 Utilizando números de sequência de 5 *bits*, qual é o tamanho máximo das janelas de envio e de recepção para cada um dos seguintes protocolos?

a. *Stop-and-Wait*

b. *Go-Back-N*

c. Repetição Seletiva

3-6 Mostre a MEF para uma máquina imaginária com três estados, A (estado inicial), B e C, além de quatro eventos, 1, 2, 3 e 4. O comportamento da máquina é especificado a seguir:

a. Quando no estado A, dois eventos podem ocorrer: evento 1 e evento 2. Se o evento 1 ocorrer, a máquina realiza a ação 1 e vai para o estado B. Se o evento 2 ocorrer, a máquina vai para o estado C (sem ação).

b. Quando no estado B, dois eventos podem ocorrer: evento 3 e evento 4. Se o evento 3 ocorrer, a máquina executa a ação 2, mas permanece no estado B. Se o evento 4 ocorrer, a máquina apenas vai para o estado C.

c. Quando no estado C, a máquina permanece nesse estado para sempre.

3-7 Considere que nossa rede nunca corrompe, perde ou duplica pacotes. Estamos preocupados apenas com o controle de fluxo. Não queremos que o emissor sobrecarregue o receptor com pacotes. Projete uma MEF para permitir que o emissor envie um pacote para o receptor apenas quando este estiver pronto. Se o receptor estiver preparado para receber um pacote, ele envia um ACK. A ausência do recebimento de um ACK pelo emissor significa que o receptor não está pronto para receber mais pacotes.

3-8 Considere que nossa rede pode corromper pacotes, mas nunca perde ou duplica um pacote. Estamos também preocupados com o controle de fluxo. Não queremos que o emissor sobrecarregue o receptor com pacotes. Projete uma MEF de um novo protocolo para suportar esses recursos.

3-9 No protocolo *Stop-and-Wait*, mostre o caso no qual o receptor recebe um pacote duplicado (que também está fora de ordem). Dica: pense em um ACK atrasado; qual é a reação do receptor frente a esse evento?

3-10 Suponha que queremos mudar o protocolo *Stop-and-Wait* e adicionar o pacote NAK (ACK negativo) ao sistema. Quando um pacote corrompido chega ao receptor, este descarta o pacote, mas envia um NAK com um nrNak igual ao nrSeq do pacote corrompido. Dessa forma, o emissor pode reenviar o pacote corrompido sem aguardar a expiração do tempo-limite. Explique quais alterações precisam ser feitas na MEF da Figura 3.21 e mostre um exemplo da operação do novo protocolo com um diagrama de sequência.

3-11 Redesenhe a Figura 3.19 com cinco pacotes (0, 1, 2, 3, 4) trocados entre o emissor e o receptor. Suponha que o pacote 2 é perdido e que o pacote 3 chega após o pacote 4.

3-12 Crie um cenário similar ao da Figura 3.22, no qual o emissor envia três pacotes. O primeiro e segundo pacotes chegam e são confirmados. O terceiro pacote é atrasado e reenviado. O pacote duplicado é recebido após a confirmação do pacote original ser enviada.

3-13 Crie um cenário similar ao da Figura 3.22, no qual o emissor envia dois pacotes. O primeiro pacote é recebido e confirmado, mas a confirmação é perdida. O emissor reenvia o pacote após a expiração do tempo-limite. O segundo pacote é perdido e reenviado.

3-14 Redesenhe a Figura 3.29 na situação em que o emissor envia cinco pacotes (0, 1, 2, 3 e 4). Os pacotes 0, 1 e 2 são enviados e confirmados com um único ACK, que chega ao emissor após todos os pacotes terem sido enviados. O pacote 3 é recebido e confirmado com um único ACK. O pacote 4 é perdido e reenviado.

3-15 Redesenhe a Figura 3.35 na situação em que o emissor envia cinco pacotes (0, 1, 2, 3 e 4). Os pacotes 0, 1 e 2 são recebidos em ordem e confirmados, um por um. O pacote 3 é atrasado, sendo recebido após o pacote 4.

3-16 Responda às seguintes questões relativas às MEFs do protocolo *Stop-and-Wait* (Figura 3.21):

a. O equipamento emissor está no estado pronto e $S = 0$. Qual é o número de sequência do próximo pacote a ser enviado?

b. O equipamento emissor está no estado bloqueado e $S = 1$. Qual é o número de sequência do próximo pacote a ser enviado caso ocorra uma expiração do tempo-limite?

c. O equipamento receptor está no estado pronto e $R = 1$. Chega um pacote com o número de sequência igual a 1. Qual é a ação em resposta a esse evento?

d. O equipamento receptor está no estado pronto e $R = 1$. Chega um pacote com o número de sequência igual 0. Qual é a ação em resposta?

3-17 Responda às seguintes questões relativas às MEFs do protocolo *Go-Back-N* com $m = 6$ *bits*. Considere que o tamanho da janela é 63. (Figura 3.27):

a. O equipamento emissor está no estado pronto com $E_p = 10$ e $E_n = 15$. Qual é o número de sequência do próximo pacote a ser enviado?

b. O equipamento emissor está no estado pronto com $E_p = 10$ e $E_n = 15$. O tempo-limite é atingido. Quantos pacotes devem ser reenviados? Quais são os seus números de sequência?

c. O equipamento emissor está no estado pronto com $E_p = 10$ e $E_n = 15$. Chega um ACK com nrAck = 13. Quais são os próximos valores de E_p e E_n?

d. O equipamento emissor está no estado bloqueado com $E_p = 14$ e $E_n = 21$. Qual é o tamanho da janela?

e. O equipamento emissor está no estado bloqueado com $E_p = 14$ e $E_n = 21$. Chega um ACK com nrAck = 18. Quais são os próximos valores de E_p e E_n? Qual é o estado do equipamento emissor?

f. O equipamento receptor está no estado pronto com $R_n = 16$. Chega um pacote com número de sequência igual a 16. Qual é o próximo valor de R_n? Qual é a resposta do equipamento para isso?

3-18 Responda às seguintes questões relativas às MEFs do protocolo de Repetição Seletiva com $m = 7$ *bits*. Considere que o tamanho da janela é 64. (Figura 3.34):

a. O equipamento emissor está no estado pronto com $E_p = 10$ e $E_n = 15$. Qual é o número de sequência do próximo pacote a ser enviado?

b. O equipamento emissor está no estado pronto com $E_p = 10$ e $E_n = 15$. O temporizador para o pacote 10 atinge o tempo-limite. Quantos pacotes devem ser reenviados? Quais são seus números de sequência?

c. O equipamento emissor está no estado pronto com $E_p = 10$ e $E_n = 15$. Chega um ACK com nrAck = 13. Quais são os próximos valores de E_p e E_n? Qual é a ação tomada em resposta a esse evento?

d. O equipamento emissor está no estado bloqueado com $E_p = 14$ e $E_n = 21$. Qual é o tamanho da janela?

e. O equipamento emissor está no estado bloqueado com $E_p = 14$ e $E_n = 21$. Chega um ACK com nrAck = 14. Os pacotes 15 e 16 já foram confirmados. Quais são os próximos valores de E_p e E_n? Qual é o estado do equipamento emissor?

f. O equipamento receptor está no estado pronto com $R_n = 16$. O tamanho da janela é 8. Chega um pacote com número de sequência igual a 16. Qual é o próximo valor de R_n? Qual é a resposta do equipamento?

3-19 Podemos definir a produto largura de banda-atraso em uma rede como o número de pacotes que podem estar no "tubo" durante o tempo de ida e volta (RTT). Qual é o produto de largura de banda-atraso em cada uma das seguintes situações?

a. Largura de banda: 1 Mbps, RTT: 20 ms, tamanho do pacote: 1.000 *bits*

b. Largura de banda: 10 Mbps, RTT: 20 ms, tamanho do pacote: 2.000 *bits*

c. Largura de banda: 1 Gbps, RTT: 4 ms, tamanho do pacote: 10.000 *bits*

3-20 Considere que precisamos projetar um protocolo *Go-Back-N* com janela deslizante para uma rede na qual a largura de banda é de 100 Mbps e a distância média entre o emissor e o receptor é 10.000 km. Considere que o tamanho médio do pacote é de 100.000 *bits* e a velocidade de propagação no meio é de 2×10^8 m/s. Determine o tamanho máximo das janelas de envio e de recepção, o número de *bits* no campo de número de sequência (m) e um valor de tempo-limite apropriado para o temporizador.

3-21 Considere que precisamos projetar um protocolo de Repetição Seletiva com janela deslizante para uma rede na qual a largura de banda é de 1 Gbps e a distância média entre o emissor e o receptor é de 5.000 km. Considere que o tamanho médio do pacote é de 50.000 *bits* e a velocidade de propagação no meio é de 2×10^8 m/s. Determine o tamanho máximo das janelas de envio e de recepção, o número de *bits* no campo de número de sequência (m) e um valor de tempo-limite apropriado para o temporizador.

3-22 Um número de confirmação no protocolo *Go-Back-N* define o próximo pacote esperado, mas um número de confirmação no protocolo de Repetição Seletiva define o número de sequência do pacote a ser confirmado. Explique o motivo.

3-23 Em uma rede usando o protocolo *Go-Back-N* com $m = 3$ e janela de envio de tamanho 7, os valores das variáveis são $E_p = 62$, $E_n = 66$ e $R_n = 64$. Considere que a rede não duplique ou reordene os pacotes.

a. Quais são os números de sequência dos pacotes de dados em trânsito?

b. Quais são os números de confirmação dos pacotes de ACK em trânsito?

3-24 Em uma rede usando o protocolo de Repetição Seletiva com $m = 4$ e janela de emissão de tamanho 8, os valores das variáveis são $E_p = 62$, $E_n = 67$ e $R_n = 64$. O pacote 65 já foi confirmado no lado do emissor; os pacotes 65 e 66 são recebidos fora de ordem no receptor. Considere que a rede não duplica os pacotes.

a. Quais são os números de sequência dos pacotes de dados pendentes (em trânsito, corrompidos ou perdidos)?

b. Quais são os números de confirmação dos pacotes de ACK pendentes (em trânsito, corrompidos ou perdidos)?

3-25 Responda às seguintes questões:

a. Qual é o tamanho mínimo de um datagrama de usuário UDP?

b. Qual é o tamanho máximo de um datagrama de usuário UDP?

c. Qual é o tamanho mínimo da carga de dados provenientes da camada de aplicação que pode ser encapsulado em um datagrama de usuário UDP?

d. Qual é o tamanho máximo da carga de dados provenientes da camada de aplicação que pode ser encapsulado em um datagrama de usuário UDP?

3-26 Um cliente usa o UDP para enviar dados para um servidor. O comprimento dos dados é de 16 *bytes*. Calcule a eficiência dessa transmissão no nível do UDP (razão entre o número de *bytes* úteis e o número total de *bytes*).

3-27 O seguinte grupo de caracteres é um *dump* (conteúdo bruto) de um cabeçalho UDP no formato hexadecimal.

0045DF0000580000

a. Qual é o número da porta de origem?

b. Qual é o número da porta de destino?

c. Qual é o tamanho total do datagrama de usuário?

d. Qual é o tamanho dos dados?

e. O pacote está indo de um cliente para um servidor ou vice-versa?

f. Qual é o protocolo de camada de aplicação?

g. O emissor calculou uma soma de verificação (*checksum*) para este pacote?

3-28 Compare o cabeçalho TCP e o cabeçalho UDP. Liste os campos no cabeçalho TCP que não fazem parte do cabeçalho UDP. Justifique a razão para cada campo ausente.

3-29 No TCP, se o valor de HLEN for 0111, quantos *bytes* de opções foram incluídos no segmento?

3-30 O que você pode dizer sobre cada um dos seguintes segmentos TCP, considerando que o valor do campo de controle é:

a. 000000 **b.** 000001 **c.** 010001

d. 000100 **e.** 000010 **f.** 010010

3-31 O campo de controle em um segmento TCP tem 6 *bits*. Podemos ter 64 diferentes combinações de *bits*. Liste algumas combinações que você acha que são normalmente utilizadas.

3-32 O seguinte grupo de caracteres é um *dump* (conteúdo bruto) de um cabeçalho TCP no formato hexadecimal.

E293 0017 00000001 00000000 5002 07FF...

- **a.** Qual é o número da porta de origem?
- **b.** Qual é o número da porta de destino?
- **c.** Qual é o número de sequência?
- **d.** Qual é o número de confirmação?
- **e.** Qual é o comprimento do cabeçalho?
- **f.** Qual é o tipo de segmento?
- **g.** Qual é o tamanho da janela?

3-33 Para entender melhor a necessidade do estabelecimento de conexão por meio do *three-way handshake* (apresentação em três vias) discutiremos um cenário. Alice e Bob não têm acesso a telefones ou à Internet (pense nos velhos tempos) para que possam combinar de fazer sua próxima reunião em um lugar distante de suas casas.

- **a.** Considere que Alice envie uma carta para Bob e especifique o dia e a hora da reunião. Alice pode ir para o local da reunião e ter certeza de que Bob estará lá?
- **b.** Considere que Bob responda ao pedido de Alice com uma carta e confirme a data e a hora. Bob pode ir para o local da reunião e ter certeza de que Alice estará lá?
- **c.** Considere que Alice responda à carta de Bob e confirme a mesma data e hora. Qualquer um dos dois pode ir ao encontro e ter a certeza de que a outra pessoa estará lá?

3-34 Para fazer com que o número de sequência inicial seja um número aleatório, a maioria dos sistemas inicia o contador em 1 durante a inicialização e incrementa o contador de 64.000 a cada meio segundo. Quanto tempo leva para o contador ser reiniciado ciclicamente?

3-35 Em uma conexão TCP, o número de sequência inicial no lado do cliente é 2.171. O cliente abre a conexão, envia três segmentos, sendo que o segundo deles carrega 1.000 *bytes* de dados, e então fecha a conexão. Qual é o valor do número de sequência em cada um dos seguintes segmentos enviados pelo cliente?

- **a.** Segmento de SYN
- **b.** Segmento de dados
- **c.** Segmento de FIN

3-36 Em uma conexão, o valor de *cwnd* é 3.000 e o valor de *rwnd* é 5.000. A estação enviou 2.000 *bytes*, que não foram confirmados. Quantos *bytes* ainda podem ser enviados?

3-37 Um cliente usa TCP para enviar dados para um servidor. Os dados consistem em 16 *bytes*. Calcule a eficiência dessa transmissão no nível do TCP (razão entre o número de *bytes* úteis e o número total de *bytes*).

3-38 Um TCP está enviando dados a 1 *megabyte* por segundo. Se o número de sequência começar em 7.000, quanto tempo demora para que o número de sequência volte a ser zero?

3-39 Um cliente HTTP abre uma conexão TCP com um número de sequência inicial (ISN) de 14.534 e cujo número de porta efêmera é 59.100. O servidor abre a conexão com um ISN de 21.732. Mostre os três segmentos TCP durante o estabelecimento da conexão supondo que o valor de *rwnd* no cliente é 4000 e o valor de *rwnd* no servidor é de 5000. Ignore o cálculo do campo de soma de verificação (*checksum*).

3-40 Considere que o cliente HTTP do problema anterior envie uma solicitação de 100 *bytes*. O servidor responde com um segmento de 1.200 *bytes*. Mostre os conteúdos dos dois segmentos trocados entre o cliente e o servidor. Suponha que a confirmação da resposta seja feita posteriormente pelo cliente. Ignore o cálculo do campo de soma de verificação (*checksum*).

3-41 Considere que o cliente HTTP do problema anterior feche a conexão e, ao mesmo tempo, confirme os *bytes* recebidos na resposta do servidor. Depois de receber o segmento de FIN do cliente, o servidor também fecha a conexão na outra direção. Mostre a fase de encerramento da conexão.

3-42 Cite as diferenças entre um evento de expiração do tempo-limite e um evento de recebimento de três ACKs duplicados. Qual deles é um forte sinal de congestionamento na rede? Por quê?

3-43 A Figura 3.52 mostra o cliente e o servidor no diagrama de transição para um cenário

comum em que é usado um fechamento do tipo *four-way handshake*. Altere o diagrama para mostrar o fechamento do tipo *three-way handshake*.

3-44 Eva, um intruso, envia um segmento de SYN para Bob, o servidor, usando o endereço IP de Alice. Eva pode criar uma conexão TCP com Bob, fingindo que ela é Alice? Considere que Bob use um ISN diferente para cada conexão.

3-45 Eva, um intruso, envia um datagrama de usuário para Bob, o servidor, usando o endereço IP de Alice. Eva pode, fingindo ser Alice, receber a resposta de Bob?

3-46 Considere que Alice, a cliente, crie uma conexão com Bob, o servidor. Eles trocam dados e fecham a conexão. Alice, então, inicia uma nova conexão com Bob, enviando um novo segmento de SYN. Antes que Bob responda a esse segmento de SYN, uma cópia duplicada do antigo segmento de SYN de Alice, que está vagando na rede, chega ao computador de Bob, disparando uma resposta dele com um segmento de SYN + ACK. Esse segmento pode ser confundido no computador de Alice como a resposta para o novo segmento de SYN? Explique.

3-47 Considere que Alice, a cliente, crie uma conexão TCP com Bob, o servidor. Eles trocam dados e fecham a conexão. Alice, então, inicia uma nova conexão com o Bob enviando um novo segmento de SYN. O servidor responde com o segmento de SYN + ACK. No entanto, antes que Bob receba de Alice o ACK para essa conexão, um antigo segmento de ACK duplicado de Alice chega a Bob. Esse ACK antigo pode ser confundido com o segmento de ACK que Bob está esperando de Alice?

3-48 Usando a Figura 3.56, explique como o controle de fluxo pode ser feito no lado do TCP emissor (do TCP emissor para a aplicação emissora). Desenhe uma representação dessa solução.

3-49 Usando a Figura 3.56, explique como o controle de fluxo pode ser feito no lado do TCP receptor (do TCP emissor para a aplicação emissora).

3-50 No TCP, considere que um cliente tenha 100 *bytes* para enviar. O cliente cria 10 *bytes* de cada vez a cada 10 ms e os entrega à camada de transporte. O servidor confirma cada segmento imediatamente ou se um temporizador expirar em 50 ms. Mostre os segmentos e os *bytes* que cada segmento carrega se a aplicação usar o algoritmo de Nagle com um tamanho máximo de segmento (MSS) de 30 *bytes*. O tempo de ida e volta (RTT) é de 20 ms, mas o temporizador do emissor está configurado para 100 ms. Todos os segmentos atingem o tamanho máximo de segmento? O algoritmo de Nagle é de fato eficaz aqui? Por quê?

3-51 Para uma visão mais clara do algoritmo de Nagle, repetiremos o problema anterior, mas deixaremos que a camada de transporte do servidor confirme um segmento mesmo quando houver um segmento anterior que não foi confirmado (qualquer outro segmento) ou se o temporizador expirar após 60 ms. Mostre o diagrama de sequência para esse cenário.

3-52 Como já explicado no texto, a janela deslizante do TCP, quando utilizada sem a nova opção de SACK, é uma combinação dos protocolos *Go-Back-N* e Repetição Seletiva. Explique quais os aspectos da janela deslizante do TCP são mais próximos do protocolo *Go-Back-N* e quais aspectos são mais próximos do protocolo de Repetição Seletiva.

3-53 Embora as novas implementações do TCP usem a opção de SACK para informar a ocorrência de *bytes* fora de ordem e de faixas duplicadas de *bytes*, explique como as implementações antigas podem sinalizar que os *bytes* em um segmento recebido estão fora de ordem ou duplicados.

3-54 Discutimos a nova opção de SACKs para o TCP no *site* do livro, mas neste momento considere que adicionamos uma opção de 8 *bytes* de NAK ao final do segmento TCP, que pode conter dois números de sequência de 32 *bits*. Mostre como podemos usar esse NAK de 8 *bytes* para sinalizar o recebimento de *bytes* fora de ordem ou de faixas de *bytes* duplicadas.

3-55 Em uma conexão TCP, considere que o tamanho máximo do segmento (MSS) seja de mil *bytes*. O processo-cliente tem 5.400 *bytes* para enviar para o processo-servidor, que não tem *bytes* para responder (comunicação unidirecional). O servidor TCP gera ACKs de acordo com as regras que discutimos no texto. Mostre a linha de tempo para as transações durante a fase de partida lenta, indicando o valor de *cwnd* no início, no fim e após cada mudança. Considere que cada cabeçalho do segmento possua apenas 20 *bytes*.

3-56 O valor de *ssthresh* para uma estação TCP Taho é de 6 MSS. A estação está no estado de partida lenta com *cwnd* 4 MSS. Mostre os valores de *cwnd* e de *ssthresh*, e o estado da estação antes e depois de cada uma das seguintes ações: chegam quatro ACKs não duplicados consecutivos, seguidos por uma expiração do tempo limite e seguidos pela chegada de três ACKs não duplicados.

3-57 O valor de *ssthresh* para um estação TCP Reno é de 8 MSS. A estação está no estado de partida lenta com *cwnd* 5 MSS e *ssthresh* = 8 MSS. Mostre os valores de *cwnd* e de *ssthresh* e o estado atual e seguinte da estação após os eventos a seguir: chegam três ACKs consecutivos não duplicados, seguidos por cinco ACKs duplicados, seguidos por dois ACKs não duplicados e seguidos por uma expiração do tempo-limite.

3-58 Em uma conexão TCP, o tamanho da janela varia entre 60.000 e 30.000 *bytes*. Se o RTT médio é de 30 ms, qual é a taxa de transferência de dados da conexão?

3-59 Se, originalmente, RRT_S 14 ms e α é ajustado para 0,2, calcule os novos RTT_S após os seguintes eventos (instantes em relação ao evento 1):

Evento 1: 00 ms Segmento 1 foi enviado.
Evento 2: 06 ms Segmento 2 foi enviado.
Evento 3: 16 ms Segmento 1 expirou o tempo-limite e foi reenviado.
Evento 4: 21 ms Segmento 1 foi confirmado.
Evento 5: 23 ms Segmento 2 foi confirmado.

3-60 Em uma conexão, suponha que o antigo $RTT_D = 7$ ms. Se o novo $RTT_S = 17$ e o novo RTT_M 20, qual é o novo valor de RTT_D? Considere $\beta = 0,25$.

3.7 EXPERIMENTOS DE SIMULAÇÃO

3.7.1 *Applets*

Criamos alguns *applets* Java para demonstrar alguns dos principais conceitos discutidos neste capítulo. Recomendamos que o estudante ative esses *applets* no *site* www.mhhe.com/forouzan e examine cuidadosamente os protocolos em ação.

3.7.2 Experimentos de laboratório

Neste capítulo, usamos Wireshark para capturar e investigar alguns pacotes da camada de transporte. Usamos o Wireshark para simular dois protocolos: UDP e TCP.

Lab3-1 No primeiro laboratório, usamos o Wireshark para capturar pacotes UDP em ação. Verificamos o valor de cada campo e se a soma de verificação (*checksum*) foi calculada para o UDP.

Lab3-2 No segundo laboratório, usamos o Wireshark para capturar e estudar em detalhe muitas das características dos pacotes TCP. Essas características incluem confiabilidade, controle de congestionamento e controle de fluxo. O Wireshark nos permite ver como o TCP usa os números de sequência e de confirmação nos segmentos para conseguir uma transferência de dados confiável. Também podemos observar os algoritmos de congestionamento do TCP (partida lenta, prevenção de congestionamento e recuperação rápida) em ação. Outra característica do TCP é o controle de fluxo. Vemos como o controle de fluxo, na direção do receptor para o emissor TCP, é obtido usando o valor de *cwnd* anunciado pelo receptor.

3.8 TAREFAS DE PROGRAMAÇÃO

Escreva o código-fonte, compile e teste os seguintes programas na linguagem de programação de sua preferência:

Prg3-1 Escreva um programa para simular a MEF para o protocolo simples no lado do emissor (Figura 3.18).

Prg3-2 Escreva um programa para simular a MEF para o protocolo *Stop-and-Wait* no lado do emissor (Figura 3.21).

Prg3-3 Escreva um programa para simular a MEF para o protocolo *Go-back-N* no lado do emissor (Figura 3.27).

Prg3-4 Escreva um programa para simular a MEF para o protocolo de Repetição Seletiva no lado do emissor (Figura 3.34).

CAPÍTULO

4 CAMADA DE REDE

A camada de rede da pilha de protocolos TCP/IP é responsável pela entrega *host* a *host* das mensagens. Esse processo de entrega inclui muitas questões, tais como o projeto de endereços lógicos que definem univocamente qualquer máquina conectada à Internet. Também são necessários alguns protocolos de roteamento que ajudem os pacotes da camada de rede a encontrar o seu caminho da origem até o destino. Neste capítulo, vamos abordar todas essas questões e combiná-las para explicar a camada de rede, discutida em cinco seções:

- Na primeira seção, apresentamos os conceitos gerais e os problemas enfrentados pela camada de rede. Inicialmente, discutimos os serviços que podem ser fornecidos na camada de rede, incluindo empacotamento, roteamento e encaminhamento. Em seguida, apresentamos a comutação de pacotes, além de questões de desempenho, congestionamento e estruturas de roteadores na camada de rede.
- Na segunda seção, explicamos a camada de rede da pilha de protocolos TCP/IP e discutimos alguns dos protocolos usados na sua versão 4, incluindo o Protocolo Internet versão 4 (IPv4 – Internet Protocol version 4) e o Protocolo de Mensagens de Controle da Internet versão 4 (ICMPv4 – Internet Control Message Protocol version 4). Também discutimos o endereçamento IPv4 e questões afins.
- Na terceira seção, abordamos o roteamento *unicast* e os protocolos de roteamento *unicast*. Mostramos como protocolos de roteamento intradomínio e interdomínios são utilizados na Internet atual.
- Na quarta seção, passamos para os protocolos de roteamento *multicast* e mostramos como alguns protocolos de roteamento *unicast* intradomínio podem ser estendidos para que sejam usados como protocolos de roteamento *multicast*. Discutimos também um novo protocolo independente de *multicast* e apresentamos brevemente o protocolo de roteamento *multicast* interdomínios.
- Na quinta seção, apresentamos a nova geração de protocolos da camada de rede, o IPv6 e o ICMPv6, bem como o endereçamento usado nessa nova geração. Deixamos a discussão sobre eles para a última seção, e o leitor pode ignorá-la sem que isso afete a compreensão do restante do material. Acreditamos que a plena implementação desses protocolos ainda levará alguns anos.

4.1 INTRODUÇÃO

A Figura 4.1 mostra a comunicação entre Alice e Bob na camada de rede. É o mesmo cenário utilizado nos Capítulos 2 e 3 para ilustrar a comunicação nas camadas de aplicação e de transporte, respectivamente.

A figura mostra que a Internet é composta por muitas redes conectadas por meio de dispositivos de conexão. Em outras palavras, é uma rede de redes, uma combinação de LANs e WANs. Para entender

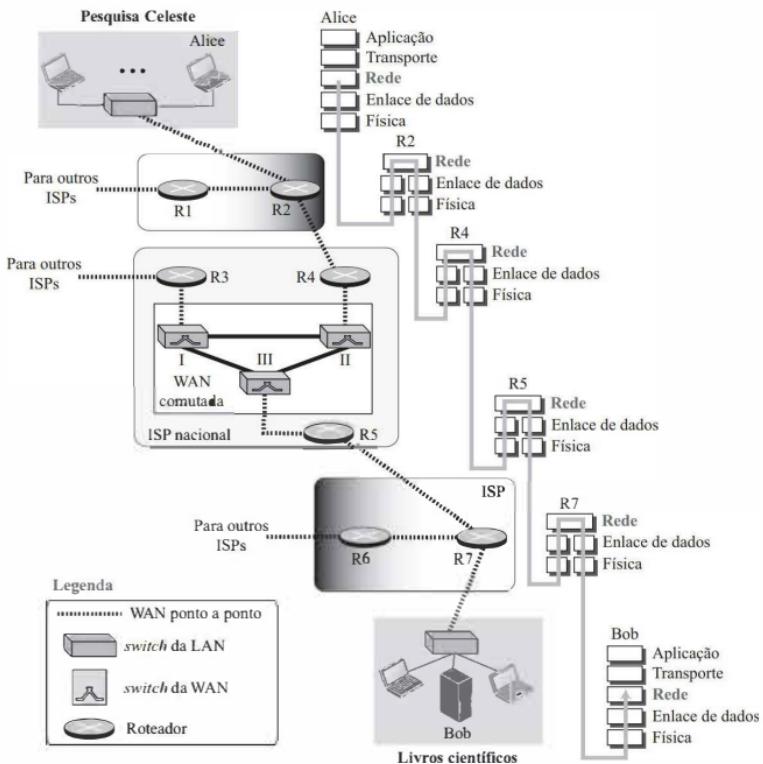



Figura 4.1 Comunicação na camada de rede.

melhor o papel da camada de rede (ou camada de inter-redes), precisamos levar em consideração os dispositivos de conexão (roteadores e *switches*) que conectam as LANs e as WANs.

Conforme mostra a Figura 4.1, a camada de rede atua nas estações de origem e de destino e em todos os roteadores no caminho entre eles (R2, R4, R5 e R7). Na estação de origem (Alice), a camada de rede recebe um pacote da camada de transporte, encapsula o pacote em um datagrama e entrega o resultado para a camada de enlace de dados. Na estação de destino (Bob), o datagrama é desencapsulado e o pacote é extraído e entregue à camada de transporte correspondente. Embora todas as cinco camadas da pilha TCP/IP atuem nas estações de origem e destino, os roteadores usam somente três camadas se eles estiverem apenas roteando pacotes; no entanto, podem precisar das camadas de transporte e aplicação para finalidades de controle. Um roteador no caminho entre duas estações costuma ser mostrado com duas camadas de enlace de dados e duas físicas, porque ele recebe um pacote de uma rede e o entrega para outra rede.

4.1.1 Serviços da camada de rede

Antes de discutir a camada de rede na Internet atual, abordaremos brevemente os serviços de rede que, em geral, são esperados de um protocolo da camada de rede.

Empacotamento

A principal tarefa da camada de rede é definitivamente o **empacotamento**: o encapsulamento da carga útil (os dados recebidos da camada superior, também chamados de *payload*) em um pacote da camada de rede na origem e o desencapsulamento da carga útil do pacote da camada de rede no destino. Em outras palavras, um dever da camada de rede é transportar uma carga útil da origem até o destino sem alterá-la ou utilizá-la. A camada de rede desempenha o papel de transportadora, assim como uma agência dos Correios, que é responsável pela entrega de pacotes de um remetente a um destinatário sem alterar ou usar o conteúdo deles.

A estação de origem recebe a carga útil de um protocolo da camada superior, adiciona um cabeçalho que contém os endereços de origem e destino e outras informações exigidas pelo protocolo da camada de rede (conforme discutido mais adiante) e entrega o pacote para a camada de enlace de dados. A origem não tem permissão de alterar o conteúdo da carga útil a não ser que ela seja muito grande para ser enviada e precise ser fragmentada.

A estação de destino recebe o pacote da camada de rede vindo de sua camada de enlace de dados, desencapsula o pacote e entrega a carga útil para o protocolo da camada superior correspondente. Se o pacote for fragmentado na origem ou por roteadores ao longo do caminho, a camada de rede é responsável por esperar até que todos os fragmentos cheguem, reorganizá-los e entregá-los ao protocolo da camada superior.

Os roteadores no caminho não têm permissão para desencapsular os pacotes que receberam, a não ser que os pacotes precisem ser fragmentados. Os roteadores também não têm permissão para alterar os endereços de origem e de destino; apenas inspecionam os endereços com o propósito de encaminhar o pacote para a próxima rede no caminho. No entanto, se um pacote estiver fragmentado, o cabeçalho precisa ser copiado para todos os fragmentos e algumas mudanças são necessárias, como discutiremos em detalhes mais adiante.

Roteamento

Outra tarefa da camada de rede, tão importante quanto a primeira, é o roteamento. A camada de rede é responsável pelo encaminhamento do pacote de sua origem até seu destino. A rede física é uma combinação de redes (LANs e WANs) e dos roteadores que as conectam. Isto significa que existe mais do que uma rota possível da origem até o destino. A camada de rede é responsável por encontrar a *melhor* entre essas possíveis rotas, e precisa adotar algumas estratégias específicas para determinar a melhor rota. Na Internet atual, isso é feito por meio da execução de alguns *protocolos de roteamento* que ajudam os roteadores a coordenar seus conhecimentos sobre suas vizinhanças e também a criar tabelas consistentes para serem usadas quando um pacote chega. Os protocolos de roteamento, que discutiremos mais adiante neste capítulo, devem ser executados antes que qualquer comunicação seja iniciada.

Encaminhamento

Se o roteamento consiste em aplicar estratégias e em executar alguns protocolos de roteamento para criar as tabelas de decisão em cada roteador, o *encaminhamento* pode ser definido como a ação tomada por cada roteador quando um pacote chega a uma de suas interfaces. A tabela de tomada de decisão de um roteador, normalmente usada para definir essa ação, é conhecida como *tabela de encaminhamento* ou *tabela de roteamento*. Quando um roteador recebe um pacote de uma das redes às quais ele está conectado, precisa encaminhá-lo a outra rede à qual ele está conectado (no caso do

roteamento *unicast*) ou para algumas redes às quais ele está conectado (no roteamento *multicast*). Para tomar essa decisão, o roteador usa parte da informação presente no cabeçalho do pacote, que pode ser seu endereço de destino ou um rótulo, com o objetivo de determinar o número da interface de saída correspondente na tabela de roteamento. A Figura 4.2 ilustra a ideia por trás do processo de encaminhamento em um roteador.

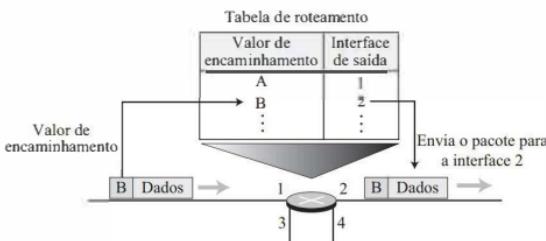



Figura 4.2 Processo de encaminhamento.

Controle de erros

No Capítulo 3, discutiremos mecanismos de controle de erros usados na camada de transporte. No Capítulo 5, explicaremos mecanismos de controle de erros na camada de enlace de dados. Embora o controle de erros também possa ser implementado na camada de rede, os projetistas ignoraram essa questão para os dados sendo transportados pela camada de rede. Uma razão para essa decisão é o fato de que o pacote na camada de rede pode ser fragmentado em cada roteador, o que torna a verificação de erros nessa camada ineficiente.

Os projetistas da camada de rede adicionaram, no entanto, um campo de soma de verificação (*checksum*) ao datagrama com o intuito de controlar qualquer tipo de corrupção no cabeçalho, mas não no datagrama inteiro. Essa verificação pode prevenir alterações ou corrupções no cabeçalho do datagrama entre dois saltos e no caminho fim a fim.

Precisamos mencionar que, embora a camada de rede na Internet não forneça controle de erros diretamente, a Internet utiliza um protocolo auxiliar, o ICMP, que de certa forma fornece controle de erros se o datagrama for descartado ou se houver alguma informação desconhecida em seu cabeçalho. Discutiremos o ICMP mais adiante.

Controle de fluxo

O controle de fluxo regula a quantidade de dados que um emissor pode enviar sem sobrecarregar o receptor. Se a camada superior no computador de origem produz dados mais rapidamente do que a camada superior no computador de destino é capaz de consumi-los, o receptor ficará sobrecarregado. Para controlar o fluxo de dados, o receptor precisa enviar alguma mensagem ao remetente para informá-lo de que a taxa de envio está excessiva.

A camada de rede na Internet, no entanto, não fornece qualquer serviço de controle de fluxo diretamente. Os datagramas são enviados pelo remetente assim que ficam prontos, sem qualquer preocupação com a capacidade do receptor.

Pode-se mencionar algumas razões para a ausência de mecanismos de controle de fluxo no projeto da camada de rede. Em primeiro lugar, como não há controle de erros nessa camada, o trabalho da camada de rede no receptor é tão simples que ele raramente acaba sobrecarregado. Em segundo lugar, as camadas superiores que usam o serviço da camada de rede podem implementar *buffers*

para receber os dados assim que eles estiverem prontos, não sendo necessário consumir os dados na mesma velocidade em que eles são recebidos. Em terceiro lugar, o controle de fluxo é fornecido para a maioria dos protocolos da camada superior que utilizam os serviços da camada de rede, de modo que outro nível de controle de fluxo faria com que a camada de rede ficasse mais complexa e tornaria o sistema inteiro menos eficaz.

Controle de congestionamento

Outra questão em um protocolo de camada de rede é o controle de congestionamento. O congestionamento é uma situação em que um número muito grande de datagramas fica concentrado em uma área da Internet. O congestionamento pode ocorrer se o número de datagramas enviados por computadores na origem for superior à capacidade da rede ou dos roteadores. Nessas ocasiões, alguns roteadores podem descartar uma parte dos datagramas. No entanto, à medida que mais datagramas são descartados, a situação pode tornar-se ainda pior porque, devido ao mecanismo de controle de erros nas camadas superiores, o remetente pode enviar cópias dos pacotes perdidos. Se o congestionamento continuar, pode-se atingir um ponto em que o sistema entra em colapso e nenhum datagrama é entregue. Discutiremos o controle de congestionamento na camada de rede mais adiante, embora ele não seja implementado na Internet.

Qualidade de serviço

À medida que a Internet passou a permitir a criação de novas aplicações, como comunicação multimídia (particularmente comunicação em tempo real de áudio e vídeo), a Qualidade de Serviço (QoS – Quality of Service) da comunicação tornou-se cada vez mais importante. A Internet prosperou devido a sua capacidade de proporcionar melhor qualidade de serviço e de dar suporte a essas aplicações. No entanto, para manter a camada de rede inalterada, tais funcionalidades são, em sua maioria, implementadas nas camadas superiores. Como a importância da QoS é mais marcante quando usamos comunicações multimídia, discutiremos esse assunto no Capítulo 8.

Segurança

Outra questão relacionada à comunicação na camada de rede é a segurança, que não era uma preocupação quando a Internet foi originalmente concebida, porque ela era usada por um pequeno número de usuários em universidades para as atividades de pesquisa; outras pessoas não tinham acesso à Internet. A camada de rede foi projetada sem funcionalidades de segurança. Hoje em dia, no entanto, a segurança é uma grande preocupação. Para garantir a segurança de uma camada de rede não orientada à conexão, é necessário haver um outro nível virtual capaz de transformar o serviço não orientado em um serviço orientado à conexão. Essa camada virtual, denominada IPSec, será discutida no Capítulo 10.

4.1.2 Comutação de pacotes

Da discussão sobre roteamento e encaminhamento na seção anterior, podemos inferir que ocorre um certo tipo de *comutação* na camada de rede. Um roteador, na realidade, atua como um interruptor (ou comutador*) que cria uma ligação entre uma porta de entrada e uma porta de saída (ou um

* N. de T.: O termo "comutador" costuma ser empregado para designar comutadores da camada de enlace, também conhecidos como *switches*. Para evitar confusões, neste livro, denominamos comutador qualquer dispositivo capaz de realizar a conexão entre uma entrada e uma saída, e reservamos o termo *switch* para os comutadores da camada de enlace.

conjunto de portas de saída), da mesma forma que um interruptor elétrico liga uma entrada a uma saída para permitir o fluxo de eletricidade.

Embora as técnicas de comutação no contexto de comunicação de dados possam ser divididas em duas grandes categorias, comutação de circuitos e comutação de pacotes, apenas esta última é utilizada na camada de redes, porque a unidade de dados nessa camada é um pacote. A comutação de circuitos é usada principalmente na camada física; o interruptor elétrico mencionado anteriormente é um tipo de comutador de circuitos.

Na camada de rede, uma mensagem da camada superior é dividida em pacotes gerenciáveis e cada pacote é enviado através da rede. A origem da mensagem envia os pacotes um a um e o destino recebe os pacotes um a um. O destino espera pela chegada de todos os pacotes que pertencem à mesma mensagem antes de entregá-la à camada superior. Os dispositivos de conexão em uma rede de comutação de pacotes ainda precisam decidir como encaminhar os pacotes para o destino final. Atualmente, uma rede de comutação de pacotes pode usar duas abordagens diferentes para encaminhar os pacotes: a *abordagem de datagramas* e a *abordagem de circuitos virtuais*. Discutiremos as duas abordagens na próxima seção.

Abordagem de datagramas: Serviço não orientado à conexão

Nos primórdios da Internet, para simplificá-la, a camada de rede foi projetada para fornecer um serviço não orientado à conexão: o protocolo da camada de rede trata cada pacote de forma independente, de modo que um pacote não tem qualquer relação com outro pacote qualquer. A ideia era que a camada de rede fosse responsável apenas pela entrega de pacotes da origem ao destino. Nesta abordagem, os pacotes de uma mensagem podem ou não percorrer o mesmo caminho até o seu destino. A Figura 4.3 ilustra essa ideia.

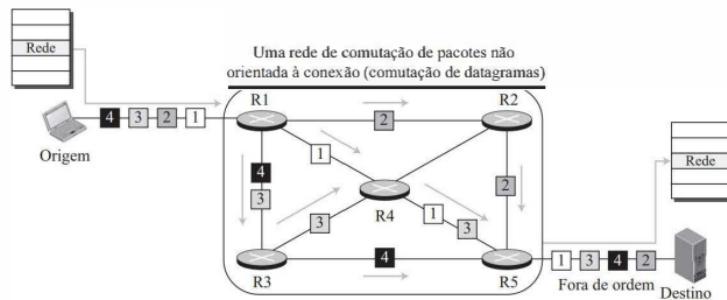



Figura 4.3 Uma rede de comutação de pacotes não orientada à conexão.

Quando a camada de rede fornece um serviço não orientado à conexão, cada pacote trafegando na Internet é uma entidade independente; não existe qualquer relação entre os pacotes pertencentes à mesma mensagem. Os comutadores nesse tipo de rede são denominados roteadores. Um pacote pertencente a uma mensagem pode ser sucedido por outro pertencente à mesma mensagem ou a uma mensagem diferente. Um pacote pode ser sucedido por outro vindo da mesma origem ou de uma diferente.

Cada pacote é roteado com base na informação contida em seu cabeçalho: os endereços de origem e de destino. O endereço de destino define aonde o pacote deve ir, enquanto o de origem define de onde ele vem. O roteador, nesse caso, encaminha o pacote com base apenas no endereço de destino. O endereço de origem pode ser usado para enviar uma mensagem de erro para a origem,

caso o pacote seja descartado. A Figura 4.4 ilustra o processo de encaminhamento em um roteador nesse caso. Usamos endereços simbólicos, tais como A e B.

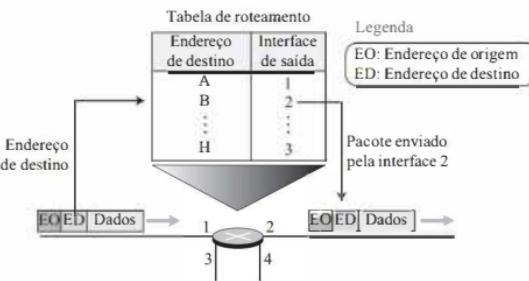



Figura 4.4 Processo de encaminhamento em um roteador em uma rede não orientada à conexão.

Na abordagem de datagramas, a decisão de encaminhamento é baseada no endereço de destino do pacote.

Abordagem de circuitos virtuais: Serviço orientado à conexão

Em um serviço orientado à conexão (também chamado de *abordagem de circuitos virtuais*), existe uma relação entre todos os pacotes que pertencem a uma mesma mensagem. Antes que todos os datagramas em uma mensagem possam ser enviados, uma conexão virtual deve ser configurada para definir o caminho a ser tomado pelos datagramas. Após a configuração da conexão, os datagramas podem seguir o mesmo caminho. Nesse tipo de serviço, o pacote não deve conter apenas os endereços de origem e destino, mas também um rótulo de fluxo, um identificador de circuito virtual que define o caminho virtual que o pacote deve seguir. Vamos mostrar em breve como esse rótulo de fluxo é determinado, mas por ora consideramos que o pacote carrega esse rótulo. Embora possa parecer que o uso desses rótulos torna o uso de endereços de origem e destino desnecessários durante a fase de transferência de dados, partes da Internet na camada de rede ainda os mantêm. Uma razão é que parte do caminho do pacote ainda pode estar usando o serviço não orientado à conexão. Outra razão é que o protocolo da camada de rede foi projetado para usar esses endereços, e pode levar algum tempo até que isso possa ser alterado. A Figura 4.5 ilustra o conceito de serviço orientado à conexão.

Cada pacote é encaminhado com base no rótulo presente no pacote. Para seguir a ideia de projeto orientado à conexão utilizado na Internet, consideramos que o pacote tem um rótulo quando chega ao roteador. A Figura 4.6 ilustra a ideia. Nesse caso, a decisão de encaminhamento é baseada no valor do rótulo, ou identificador de circuito virtual, como também é conhecido.

Para criar um serviço orientado à conexão, um processo em três fases é utilizado: configuração, transferência de dados e finalização. Na fase de configuração, os endereços de origem e destino do emissor e do receptor são usados para criar entradas na tabela usada para o serviço orientado à conexão. Na fase de finalização, a origem e o destino informam o roteador para que ele apague as entradas correspondentes. A transferência de dados ocorre entre essas duas fases.

Na abordagem de circuitos virtuais, a decisão sobre o encaminhamento é baseada no rótulo do pacote.

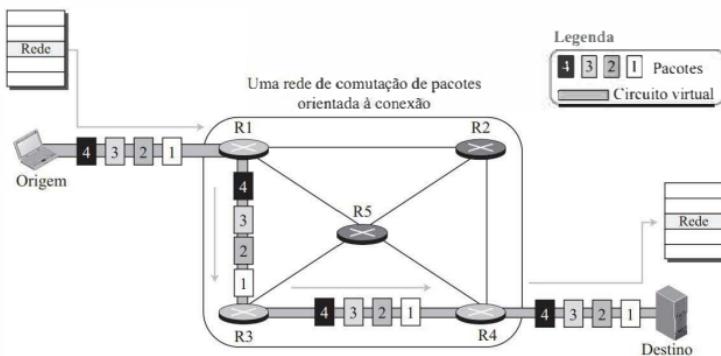



Figura 4.5 Uma rede de comutação de pacotes usando circuitos virtuais.

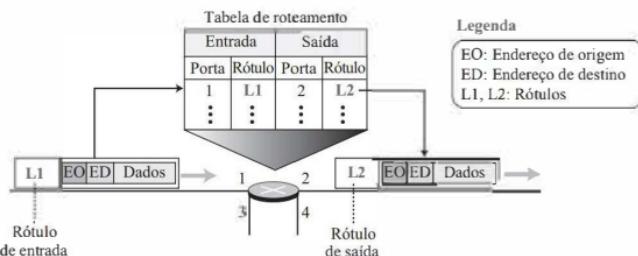



Figura 4.6 Processo de encaminhamento em um roteador quando utilizado em uma rede de circuitos virtuais.

Fase de configuração

Na fase de configuração, um roteador cria uma entrada para um circuito virtual. Por exemplo, suponha que a origem A precisa criar um circuito virtual para o destino B. Dois pacotes auxiliares precisam ser trocados entre o emissor e o receptor: o pacote de solicitação e o pacote de confirmação (*acknowledgment*).

Pacote de solicitação Um pacote de solicitação é enviado da origem para o destino. Esse pacote auxiliar carrega os endereços de origem e de destino. A Figura 4.7 ilustra o processo.

1. A origem A envia um pacote de solicitação ao roteador R1.
2. O roteador R1 recebe o pacote de solicitação. Ele sabe que um pacote indo de A para B sai pela porta 3. O processo pelo qual o roteador obtém essa informação é descrito mais adiante. Por enquanto, considere que ele sabe qual é a porta de saída. O roteador cria uma entrada na tabela para esse circuito virtual, mas é capaz de preencher apenas três das quatro colunas. O roteador preenche a porta de entrada (1) e escolhe um rótulo de entrada que esteja disponível (14) e a porta de saída (3). Ele ainda não sabe qual é o rótulo de saída, que será definido durante o passo de confirmação. O roteador, então, encaminha o pacote pela porta 3 para o roteador R3.

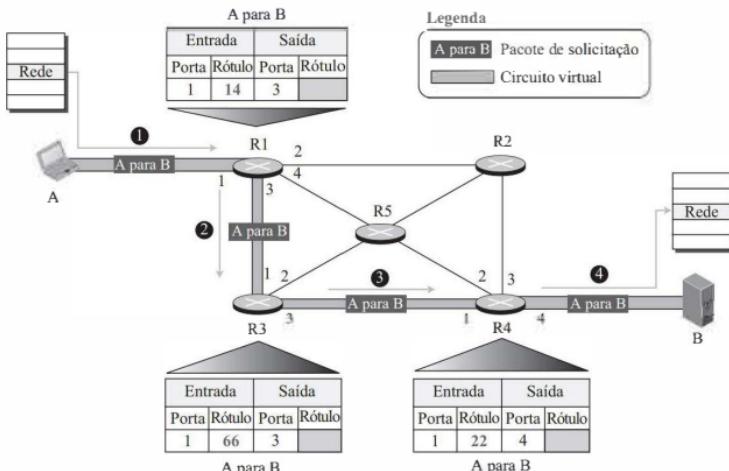



Figura 4.7 Envio de pacote de solicitação em uma rede de circuitos virtuais.

3. O roteador R3 recebe o pacote de solicitação de configuração. Os mesmos eventos que acontecem no roteador R1 também ocorrem aqui; três colunas da tabela são preenchidas: nesse caso, a porta de entrada (1), o rótulo de entrada (66) e a porta de saída (3).
4. O roteador R4 recebe o pacote de solicitação de configuração. Mais uma vez, três colunas são preenchidas: a porta de entrada (1), o rótulo de entrada (22) e a porta de saída (4).
5. O destino B recebe o pacote de configuração e, se ele estiver pronto para receber pacotes de A, atribui um rótulo para os pacotes de entrada que vêm de A, nesse caso, 77, conforme mostra a Figura 4.8. Esse rótulo permite que o destino saiba que os pacotes são provenientes de A e não de outras origens.

Pacote de confirmação Um pacote especial, chamado pacote de confirmação, completa as entradas nas tabelas de roteamento. A Figura 4.8 ilustra o processo.

1. O destino envia uma confirmação para o roteador R4. O pacote de confirmação carrega os endereços de origem e destino globais de modo que o roteador saiba qual entrada na tabela deve ser preenchida. O pacote também carrega o rótulo 77, escolhido pelo destino como o rótulo de entrada para os pacotes provenientes de A. O roteador R4 usa esse rótulo para preencher a coluna de rótulo de saída para a entrada. Observe que 77 é o rótulo de entrada para o destino B, mas é o rótulo de saída para o roteador R4.
2. O roteador R4 envia um pacote de confirmação para o roteador R3. Esse pacote carrega o rótulo de entrada da tabela de R4, escolhido na fase de configuração. O roteador R3 usa esse valor como o rótulo de saída em sua própria tabela.
3. O roteador R3 envia um pacote de confirmação para o roteador R1. Esse pacote carrega o rótulo de entrada da tabela de R3, escolhido na fase de configuração. O roteador R1 usa esse valor como o rótulo de saída em sua própria tabela.

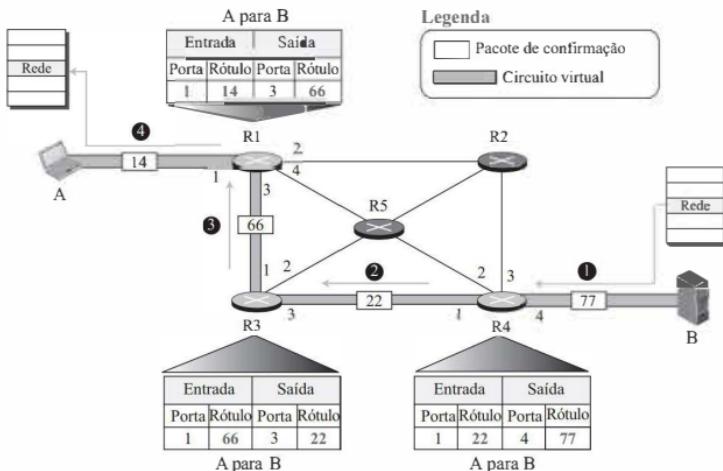



Figura 4.8 Envio de confirmações em uma rede de circuitos virtuais.

4. Finalmente, o roteador R1 envia um pacote de confirmação para a origem A. Esse pacote carrega o rótulo de entrada da tabela de R1, escolhido na fase de configuração.
5. A origem usa esse valor como o rótulo de saída para os pacotes de dados que forem enviados para o destino B.

Fase de transferência de dados

A segunda fase é chamada fase de transferência de dados. Após todos os roteadores terem criado suas tabelas de roteamento para um circuito virtual específico, os pacotes da camada de rede pertencentes a uma mensagem podem ser enviados um após o outro. A Figura 4.9 mostra o fluxo de um único pacote, mas o processo é o mesmo para 1, 2, 100 ou qualquer outro número de pacotes. O computador de origem usa o rótulo 14, que foi recebido do roteador R1 na fase de configuração. O roteador R1 encaminha o pacote para o roteador R3, mas altera o rótulo para 66. O roteador R3 encaminha o pacote para o roteador R4, mas modifica o rótulo para 22. Finalmente, o roteador R4 entrega o pacote ao seu destino final com o rótulo 77. Todos os pacotes na mensagem seguem a mesma sequência de rótulos e os pacotes chegam ordenados ao destino.

Fase de finalização

Na fase de finalização, a origem A, depois de enviar todos os pacotes para B, envia um pacote especial chamado pacote de finalização. O destino B responde com um pacote de confirmação. Todos os roteadores excluem as entradas correspondentes em suas tabelas.

4.1.3 Desempenho da camada de rede

Os protocolos das camadas superiores que utilizam os serviços da camada de rede esperam receber um serviço ideal, mas a camada de rede não é perfeita. O desempenho de uma rede pode

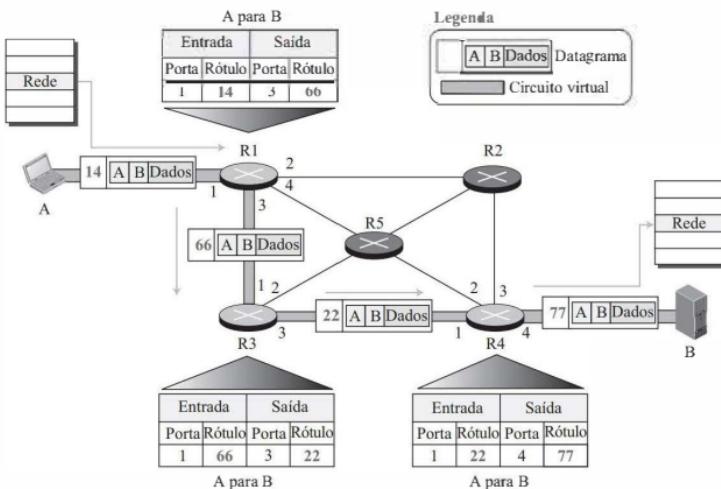



Figura 4.9 Fluxo de um pacote em um circuito virtual estabelecido.

ser medido em termos de *atraso*, *vazão* e *perda de pacotes*. Primeiramente, definimos esses três termos em uma rede de comutação de pacotes e depois discutimos seus efeitos sobre o desempenho da rede.

Atraso

Todos esperamos respostas instantâneas de uma rede, mas um pacote encontra atrasos da origem ao destino, que podem ser divididos em quatro tipos: atraso de transmissão, de propagação, de processamento e de fila. Em primeiro lugar, discutiremos todos esses tipos de atrasos e, em seguida, mostraremos como calcular o atraso dos pacotes da origem até o destino.

Atraso de transmissão

Estações de origem e roteadores não podem enviar pacotes instantaneamente. Um remetente precisa inserir os *bits* de um pacote na linha de transmissão um por um. Se o primeiro *bit* do pacote for colocado na linha no instante t_1 e o último *bit* for colocado na linha no instante t_2, o atraso de transmissão do pacote é dado por $(t_2 - t_1)$. Definitivamente, o atraso de transmissão é mais longo para um pacote maior e mais curto se o remetente for capaz de transmitir mais rápido. Em outras palavras, o atraso de transmissão é calculado como:

$$\textbf{Atraso}_{tr} = \textbf{(Tamanho do pacote) / (Taxa de transmissão).}$$

Por exemplo, em uma LAN do tipo Fast Ethernet (ver Capítulo 5) com uma taxa de transmissão de 100 milhões de *bits* por segundo e para um pacote de 10.000 de *bits*, levam-se (10.000)/(100.000.000) ou 100 microssegundos para que todos os *bits* do pacote sejam colocados na linha.

Atraso de propagação

O atraso de propagação é o tempo que um *bit* leva para viajar do ponto A ao ponto B no meio de transmissão. O atraso de propagação para uma rede de comutação de pacotes depende do atraso de propagação de cada rede (LAN ou WAN). O atraso de propagação depende da velocidade de propagação no meio de transmissão, que é dado por 3×10^8 metros/segundo no vácuo e normalmente é muito menor em um meio com fio; o atraso também depende da distância entre os pontos. Em outras palavras, o atraso de propagação é calculado como:

$$\textbf{Atraso}_{pg} = \textbf{(Distância) / (Velocidade de propagação).}$$

Por exemplo, se a distância de uma conexão por cabo em uma WAN ponto a ponto for de 2.000 metros e a velocidade de propagação dos *bits* no cabo for 2×10^8 metros/segundo, então, o atraso de propagação será de 10 microssegundos.

Atraso de processamento

O atraso de processamento é o tempo necessário para que um roteador ou uma estação de destino receba um pacote em sua porta de entrada, remova o cabeçalho do pacote, execute um procedimento de detecção de erros e entregue o pacote na porta de saída (no caso de um roteador) ou para o protocolo da camada superior (no caso da estação de destino). O atraso de processamento pode ser diferente para cada pacote, mas normalmente é calculado como uma média.

$$\textbf{Atraso}_{pr} = \textbf{Tempo necessário para processar um pacote em um roteador ou estação de destino}$$

Atraso de fila

O atraso de fila pode normalmente acontecer em um roteador. Como discutiremos na próxima seção, um roteador possui uma fila de entrada conectada a cada uma das portas de entrada para armazenar pacotes à espera de processamento; o roteador também tem uma fila de saída ligada a cada uma das portas de saída para armazenar pacotes esperando para serem transmitidos. O atraso de fila de um pacote em um roteador é medido como o tempo que um pacote aguarda nas filas de entrada e de saída. Podemos comparar essa situação a um aeroporto movimentado. Alguns aviões podem ter de esperar para conseguir uma pista de pouso (atraso de entrada), enquanto outros podem ter de esperar por uma pista de decolagem (atraso de saída).

$$\textbf{Atraso}_{fi} = \textbf{Tempo que um pacote espera nas filas de entrada e saída de um roteador}$$

Atraso total

Considerando atrasos iguais para o remetente, roteadores e receptor, o atraso total (atraso da origem ao destino) encontrado por um pacote pode ser calculado se soubermos o número de roteadores, *n*, no caminho todo.

$$\textbf{Atraso total} = (n + 1)\,(\textbf{Atraso}_{tr} + \textbf{Atraso}_{pg} + \textbf{Atraso}_{pr}) + (n)\,(\textbf{Atraso}_{fi})$$

Observe que, se temos n roteadores, temos $(n + 1)$ enlaces. Portanto, temos $(n + 1)$ atrasos de transmissão relacionados a n roteadores e à fonte, $(n + 1)$ atrasos de propagação relacionados a $(n + 1)$ enlaces, $(n + 1)$ atrasos de processamento relacionados a n roteadores e ao destino, e apenas n atrasos de fila relacionados aos n roteadores.

Vazão

A vazão em qualquer ponto de uma rede é definida como o número de *bits* que passam por um ponto em um segundo, correspondendo, na realidade, à taxa de transmissão de dados naquele ponto. Em um caminho da origem até o destino, um pacote pode passar por vários enlaces, cada um com uma

taxa de transmissão diferente. Como, então, podemos determinar a vazão do caminho completo? Para entender essa situação, considere que temos três enlaces, cada um com uma taxa de transmissão diferente, conforme mostra a Figura 4.10.

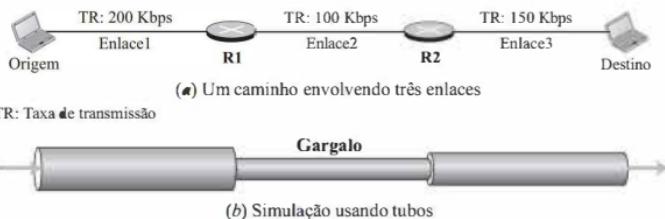



Figura 4.10 Vazão em um caminho com três enlaces encadeados.

Na figura, os dados podem trafegar a uma taxa de 200 Kbps no Enlace1. No entanto, quando os dados chegam ao roteador R1, eles não podem trafegar com esta taxa. Os dados precisam ser enfileirados no roteador e enviados a 100 Kbps. Quando chegam ao roteador R2, poderiam ser enviados a uma taxa de 150 Kbps, mas não há dados suficientes para serem enviados a essa taxa. Em outras palavras, a taxa média do fluxo de dados no Enlace3 também é 100 Kbps. Podemos concluir que a taxa de dados média para esse caminho é de 100 Kbps, o mínimo das três diferentes taxas de dados. A figura também mostra que podemos simular o comportamento de cada enlace com tubos de diferentes tamanhos; a vazão média é determinada pelo gargalo, o tubo com o menor diâmetro. Em geral, em um caminho com n enlaces em série, temos

$$\textbf{Vazão} = \textbf{mínimo}\{TT_1, TT_2, ..., TT_n\}.$$

Embora o cenário da Figura 4.10 mostre como calcular a vazão quando os dados passam através de vários enlaces, a situação real da Internet é que os dados normalmente passam por duas redes de acesso e pelo *backbone* (espinha dorsal) da Internet, conforme mostra a Figura 4.11.

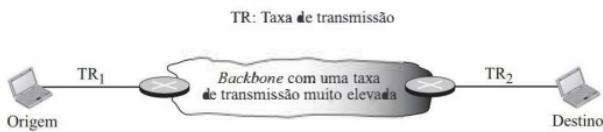



Figura 4.11 Um caminho através do *backbone* da Internet.

O *backbone* da Internet tem uma taxa de transmissão muito elevada, na faixa de *gigabits* por segundo. Isto significa que a vazão é normalmente definida como a taxa de transmissão mínima entre os dois enlaces de acesso que ligam a origem e o destino ao *backbone*. A Figura 4.11 mostra esse cenário, em que a taxa de transferência é dada pelo mínimo entre TT_1 e TT_2. Por exemplo, se um servidor está conectado à Internet por meio de uma LAN Fast Ethernet com uma taxa de transmissão de dados de 100 Mbps, mas um usuário que quer obter um arquivo está conectado à Internet por uma linha telefônica discada com uma taxa de transmissão de dados de 40 Kbps, a vazão é de 40 Kbps. O gargalo é definitivamente a linha discada.

É necessário mencionar uma outra situação em que levamos a vazão em consideração. O enlace entre dois roteadores nem sempre é dedicado a um único fluxo. Um roteador pode receber fluxos de várias fontes ou distribuí-lo a diversos destinos. Nesse caso, a taxa de transmissão do enlace entre os dois roteadores é, na verdade, compartilhada entre os fluxos, e isso deve ser levado em consideração quando calculamos a vazão. Por exemplo, na Figura 4.12, a taxa de transmissão do enlace principal no cálculo da vazão é de apenas 200 Kbps, porque o enlace é compartilhado por três fluxos.

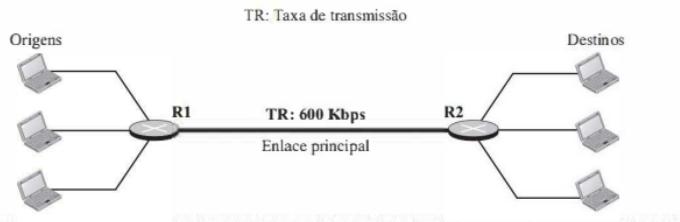



Figura 4.12 Efeito da vazão em enlaces compartilhados.

Perda de pacotes

Outro fator que afeta consideravelmente o desempenho da comunicação é o número de pacotes perdidos durante a transmissão. Quando um roteador recebe um pacote enquanto ele processa outro, o pacote recebido deve ser armazenado em um *buffer* de entrada e esperar pela sua vez. Um roteador, contudo, tem um *buffer* de entrada de tamanho limitado. Pode chegar um momento em que o *buffer* esteja cheio, de modo que o próximo pacote a chegar deve ser descartado. A consequência da perda de pacotes na camada de rede da Internet é que o pacote precisa ser reenviado, o que, por sua vez, pode criar um tráfego excessivo e causar a perda de mais pacotes. Existem diversos estudos teóricos na área da teoria de filas que buscam evitar a sobrecarga das filas e a perda de pacotes.

4.1.4 Congestionamento na camada de rede

No Capítulo 3, discutimos o congestionamento na camada de transporte. Embora o congestionamento na camada de rede não seja explicitamente abordado no modelo Internet, o estudo do congestionamento nessa camada pode nos ajudar a entender melhor a causa do congestionamento na camada de transporte e a encontrar possíveis soluções utilizáveis na camada de rede. O congestionamento na camada de rede está relacionado a duas questões, vazão e atraso, que discutimos na seção anterior.

A Figura 4.13 mostra essas duas medidas de desempenho em função da carga na rede.

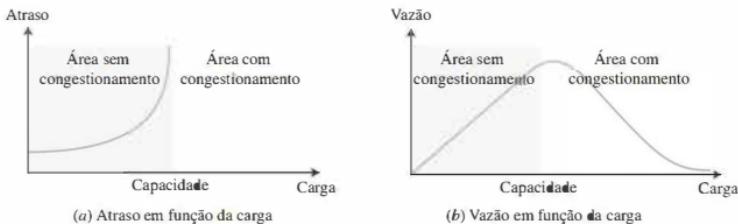



Figura 4.13 Atraso e vazão de pacotes em função da carga na rede.

Quando a carga na rede é muito menor do que sua capacidade, o **atraso** é mínimo. Esse atraso mínimo é composto pelos atrasos de propagação e de processamento, ambos pouco significativos. No entanto, quando a carga atinge a capacidade da rede, o atraso aumenta rapidamente porque nesse caso precisamos incluir o atraso das filas no cálculo do atraso total. Perceba que o atraso tende ao infinito quando a carga na rede é maior do que sua capacidade.

Quando a carga encontra-se abaixo da capacidade da rede, a sua vazão aumenta proporcionalmente à *carga*. Podemos esperar que a vazão se mantenha constante após a carga da rede atingir sua capacidade máxima, mas em vez disso, a vazão diminui drasticamente. O motivo é o descarte de pacotes pelos roteadores. Quando a carga da rede excede sua capacidade, as filas ficam cheias e os roteadores têm que descartar alguns pacotes, o que não reduz o número de pacotes na rede, porque os remetentes retransmitem os pacotes, utilizando mecanismos de temporização, quando os pacotes não chegam aos destinatários.

Controle de congestionamento

Conforme discutimos no Capítulo 3, o controle de congestionamento consiste em técnicas e mecanismos que podem tanto evitar o congestionamento antes que ele aconteça ou eliminá-lo depois de ocorrido. Em geral, podemos dividir os mecanismos de controle de congestionamento em duas grandes categorias: **controle de congestionamento em malha aberta** (*open-loop*), que é voltado à prevenção do congestionamento, e **controle de congestionamento em malha fechada** (*closed-loop*), voltado à sua eliminação.

Controle de congestionamento em malha aberta

No controle de congestionamento em malha aberta, são aplicadas políticas para evitar o congestionamento antes que ele ocorra. Nesses mecanismos, o controle de congestionamento é gerenciado pela origem ou pelo destino. Apresentamos a seguir uma breve lista de políticas que podem evitar congestionamento.

Política de retransmissão A retransmissão às vezes é inevitável. Se o remetente acreditar que um pacote enviado foi perdido ou corrompido, precisa retransmiti-lo. A retransmissão, em geral, pode aumentar o congestionamento da rede, mas uma boa política de retransmissão pode evitá-lo. A política e os temporizadores de retransmissão devem ser projetados para garantir elevada eficiência e, ao mesmo tempo, evitar congestionamentos.

Política de janela O tipo de janela no remetente também pode afetar o congestionamento. Uma janela com retransmissão seletiva (*Selective Repeat*) é melhor do que a janela com retransmissão integral (*Go-Back-N*) para controle de congestionamento. Na janela com retransmissão integral, quando o temporizador dos pacotes é disparado, vários pacotes podem ser reenviados, embora alguns deles possam já ter chegado ao receptor. Essa duplicação pode piorar o congestionamento. Uma janela com retransmissão seletiva, por outro lado, tenta enviar especificamente os pacotes que foram perdidos ou corrompidos.

Política de confirmação A política de confirmação imposta pelo receptor também pode afetar o congestionamento. Se o receptor não enviar confirmações para todos os pacotes que recebe, pode reduzir a velocidade de envio do remetente e ajudar a evitar o congestionamento. Diversas abordagens são utilizadas nesse caso. Um receptor pode enviar uma confirmação apenas se tiver um pacote para ser enviado ou após o disparo de um temporizador especial. Um receptor também pode decidir enviar uma confirmação de N pacotes de cada vez. Perceba que as confirmações também fazem parte da carga em uma rede. O envio de um menor número de confirmações impõe uma menor carga na rede.

Política de descarte Uma boa política de descarte por parte dos roteadores deve evitar o congestionamento e, ao mesmo tempo, não deve prejudicar a integridade da transmissão. Por exemplo,

em uma transmissão de áudio, se a política ditar o descarte de pacotes menos sensíveis quando a ocorrência de congestionamento for provável, a qualidade do som ainda é preservada e o congestionamento é evitado ou aliviado.

Política de admissão Uma política de admissão, que é um mecanismo usado para prover qualidade de serviço (assunto discutido no Capítulo 8), também pode evitar o congestionamento em redes de circuitos virtuais. Comutadores envolvidos em um fluxo primeiramente verificam os recursos exigidos por tal fluxo antes de admiti-lo na rede. Um roteador pode recusar o estabelecimento de uma conexão baseada em circuito virtual se houver congestionamento na rede ou alguma possibilidade de congestionamento no futuro.

Controle de congestionamento em malha fechada

Mecanismos de controle de congestionamento em malha fechada (*closed-loop*) tentam aliviar o congestionamento depois que ele acontece. Diversos mecanismos têm sido usados por diferentes protocolos. Descrevemos alguns deles aqui.

Contrapressão A técnica de *contrapressão* (*backpressure*) refere-se a um mecanismo de controle de congestionamento no qual um nó congestionado deixa de receber dados vindos do nó ou nós no sentido inverso ao fluxo. Isto pode fazer com que o nó ou nós anteriores fiquem congestionados e, por sua vez, rejeitem dados de nó(s) anterior(es) a ele e assim por diante. A contrapressão é um mecanismo de controle de congestionamento nó a nó, que começa com um nó e propaga-se no sentido oposto do fluxo de dados, até a origem. A técnica de contrapressão pode ser aplicada apenas às redes baseadas em circuitos virtuais, nas quais cada nó conhece o nó anterior, de onde um fluxo de dados está vindo. A Figura 4.14 ilustra a ideia da contrapressão.

Figura 4.14 Método de contrapressão para aliviar congestionamento.

O nó III na figura está recebendo mais dados em sua entrada do que ele é capaz de tratar. Ele descarta alguns pacotes do seu *buffer* de entrada e pede ao nó II que reduza a taxa de envio de dados. O nó II, por sua vez, pode ficar congestionado porque está reduzindo o fluxo de saída de dados. Se o nó II ficar congestionado, ele pede ao nó I que reduza a taxa de envio, o que pode criar congestionamento no nó I. Se isso acontecer, esse nó pede à origem que reduza a taxa de envio de dados. Isso, após algum tempo, alivia o congestionamento. Perceba que a *pressão* no nó III é movida no sentido inverso ao fluxo, até a origem, para remover o congestionamento.

É importante mencionar que esse tipo de controle de congestionamento só pode ser implementado em redes de circuitos virtuais, e não em uma rede de datagramas, na qual um nó (roteador) não tem qualquer conhecimento sobre qual é o roteador de onde vem o fluxo.

Pacote de bloqueio Um **pacote de bloqueio** (*choke packet*) é um pacote enviado por um nó para a origem dos dados com o intuito de informá-la a respeito de um congestionamento. Perceba a diferença entre os métodos de contrapressão e pacote de bloqueio. Na contrapressão, o aviso vai de um nó para o nó imediatamente anterior no fluxo, embora o aviso possa, eventualmente, chegar à estação de origem. No método dos pacotes de bloqueio, o aviso provém do roteador que encontrou o congestionamento, e vai diretamente para a estação de origem. Os nós intermediários pelos quais o pacote passa não são avisados. Veremos um exemplo desse tipo de controle

no ICMP (discutido mais adiante). Quando um roteador na Internet se vê sobrecarregado de datagramas IP, ele pode descartar alguns deles, mas informa a estação de origem usando uma mensagem de ICMP do tipo *origem extinta*. A mensagem de aviso vai diretamente para a estação de origem; os roteadores intermediários não atuam de forma alguma. A Figura 4.15 ilustra a ideia de um pacote de bloqueio.

Figura 4.15 Pacote de bloqueio.

Sinalização implícita Na sinalização implícita, não há qualquer comunicação entre o(s) nó(s) congestionado(s) e a origem. A origem acredita que há congestionamento em algum lugar na rede devido a outros sintomas. Por exemplo, quando uma estação de origem envia vários pacotes e não recebe confirmações por algum tempo, uma hipótese é a de que a rede esteja congestionada. A demora em receber confirmações é interpretada como um congestionamento na rede e a fonte deve reduzir sua taxa de envio. Vimos esse tipo de sinalização quando discutimos o controle de congestionamento do TCP no Capítulo 3.

Sinalização explícita O nó que sofre congestionamento pode explicitamente enviar um sinal para a origem ou destino. O método de sinalização explícita, no entanto, é diferente do método de pacote de bloqueio. Neste, um pacote separado é usado para isso; no método explícito de sinalização, o sinal é incluído nos pacotes que transportam dados. A sinalização explícita pode ocorrer em qualquer direção do tráfego, ou seja, da origem para o destino ou vice-versa. Esse tipo de controle de congestionamento poderá ser visto em uma rede ATM, discutida no Capítulo 5.

4.1.5 Estrutura de um roteador

Em nossa discussão sobre encaminhamento e roteamento, representamos um roteador como uma caixa preta que aceita os pacotes que chegam em uma das portas (interfaces) de entrada, utiliza uma tabela de roteamento para encontrar a porta de saída pela qual o pacote deve sair e então envia o pacote por ela. Nesta seção, abriremos a caixa preta e olharemos dentro dela. No entanto, nossa discussão não será muito detalhada; livros inteiros foram escritos sobre roteadores. Fornecemos apenas uma visão geral para o leitor.

Componentes

Pode-se dizer que um roteador tem quatro componentes: *portas de entrada*, *portas de saída*, o *processador de roteamento* e a *malha de comutação*, conforme mostra a Figura 4.16.

Portas de entrada

A Figura 4.17 mostra um diagrama esquemático de uma porta de entrada.

Figura 4.16 Componentes de um roteador.

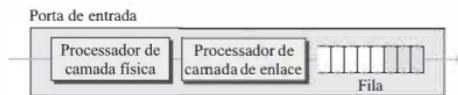



Figura 4.17 Porta de entrada.

Uma porta de entrada executa as funções das camadas física e de enlace do roteador. Os *bits* são construídos a partir do sinal recebido. O pacote é desencapsulado a partir do quadro, erros são verificados e o pacote é descartado se estiver corrompido; então, estará pronto para ser processado pela camada de rede. Além dos processadores da camada física e de enlace, a porta de entrada tem *buffers* (filas) para armazenar temporariamente os pacotes antes que eles sejam direcionados para a malha de comutação.

Portas de saída

Uma porta de saída executa as mesmas funções que uma porta de entrada, mas na ordem inversa. Primeiro, os pacotes de saída são colocados em uma fila, e cada um é encapsulado em um quadro e, finalmente, as funções da camada física são aplicadas ao quadro para criar o sinal a ser colocado na linha. A Figura 4.18 mostra um diagrama esquemático de uma porta de saída.

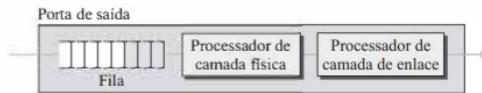



Figura 4.18 Porta de saída.

Processador de roteamento

O processador de roteamento executa as funções da camada de rede. O endereço de destino é usado para determinar o endereço do próximo salto e, ao mesmo tempo, o número da porta de saída pela qual o pacote será enviado. Essa atividade é também conhecida como *table lookup* (consulta à tabela) porque o processador de roteamento faz uma busca na tabela de roteamento. Nos roteadores mais novos, essa função do processador de roteamento geralmente é transferida para as portas de entrada para facilitar e agilizar o processo.

Malhas de comutação

A tarefa mais difícil em um roteador é mover o pacote da fila de entrada para a de saída. A velocidade com que isso é feito afeta o tamanho das filas de entrada e saída e também o atraso global na

entrega de pacotes. Antigamente, quando um roteador era um computador dedicado, sua memória – ou um barramento – era usada como malha de comutação. A porta de entrada armazenava o pacote na memória e, posteriormente, o pacote era lido pela porta de saída. Hoje em dia, os roteadores usam várias malhas de comutação; discutiremos algumas delas a seguir.

Comutador *crossbar* O tipo mais simples de malha de comutação é a *crossbar* (barras cruzadas), mostrado na Figura 4.19. Um **comutador *crossbar*** conecta *n* entradas a *n* saídas em uma matriz, usando microinterruptores eletrônicos em cada **ponto de cruzamento** (*crosspoint*).

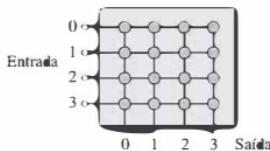



Figura 4.19 Comutador *crossbar.*

Comutador *banyan* Mais realista que o comutador *crossbar* é o comutador *banyan* (nomeado em referência à árvore figueira, ou *banyan tree*, em inglês), conforme mostra a Figura 4.20.

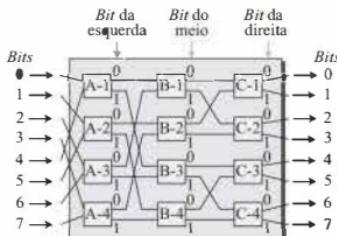
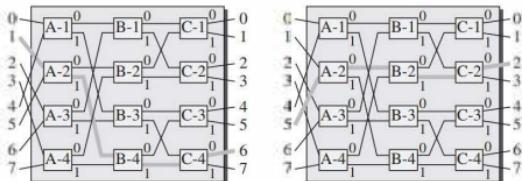



Figura 4.20 Comutador *banyan.*

Um comutador *banyan* é um comutador multiestágios com vários microcomutadores em cada estágio. Ele encaminha os pacotes com base na porta de saída, representada como uma sequência binária. Para *n* entradas e *n* saídas, temos $\log_2(n)$ estágios com *n*/2 microcomutadores em cada estágio. O primeiro estágio encaminha o pacote com base no *bit* mais significativo da sequência binária. O segundo estágio encaminha os pacotes com base no segundo *bit* mais significativo, e assim por diante. A Figura 4.21 mostra um comutador *banyan* com oito entradas e oito saídas. O número de estágios é $\log_2(8) = 3$. Considere que um pacote chegou na porta de entrada 1 e deve ir para a porta de saída 6 (110 em binário), conforme mostra a parte (*a*) da figura. O primeiro microcomutador (A-2) encaminha o pacote com base no primeiro *bit* (1); o segundo microcomutador (B-4) encaminha o pacote com base no segundo *bit* (1); o terceiro microcomutador (C-4) encaminha o pacote com base no terceiro *bit* (0). Na parte (*b*) da figura, um pacote chega na porta de entrada 5 e deve ir para a porta de saída 2 (010 em binário). O primeiro microcomutador (A-2) encaminha o pacote com base no primeiro *bit* (0); o segundo microcomutador (B-2) encaminha o pacote com base no segundo *bit* (1); o terceiro microcomutador (C-2) encaminha o pacote com base no terceiro *bit* (0).

(*a*) Entrada 1 enviando para saída 6 (110) (*b*) Entrada 5 enviando para saída 2 (010)

Figura 4.21 Exemplos de roteamento em um comutador *banyan.*

Comutador Batcher-banyan O problema com o **comutador *banyan*** é a possibilidade de colisões internas, mesmo quando dois pacotes não estão indo para a mesma porta de saída. Podemos resolver esse problema classificando os pacotes que chegam com base em sua porta de destino. K. E. Batcher desenvolveu um comutador que pode ser colocado antes do comutador *banyan* e ordena os pacotes entrantes de acordo com seu destino final. A combinação é denominada comutador Batcher-banyan (ver Figura 4.22).

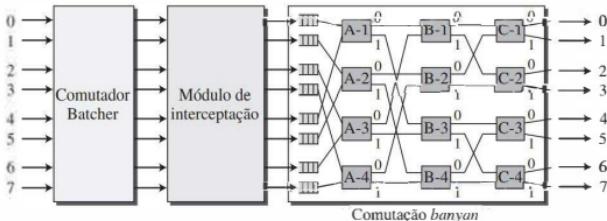



Figura 4.22 Comutador Batcher-banyan.

O comutador de ordenação usa técnicas de mesclagem de dados por *hardware*, mas não discutiremos os detalhes aqui. Normalmente, um outro módulo de *hardware*, chamado *módulo de interceptação* (*trap*), é adicionado entre o comutador Batcher e o comutador *banyan*. O módulo de interceptação evita que os pacotes duplicados (aqueles que tenham a mesma saída como destino) passem para o comutador *banyan* simultaneamente. A cada pulso de um relógio, um único pacote para cada destino é tratado; se houver mais de um pacote para o mesmo destino, os outros devem esperar pelo próximo pulso do relógio.

4.2 PROTOCOLOS DA CAMADA DE REDE

Na primeira seção deste capítulo, discutimos questões gerais e os serviços associados à camada de rede. Nesta seção, mostraremos como a camada de rede é implementada na pilha de protocolos TCP/IP. Os protocolos da camada de rede passaram por diversas versões; nesta seção, nos concentraremos na versão atual (a versão 4), enquanto na última seção deste capítulo, discutiremos brevemente a versão 6, que ainda não foi completamente implementada.

A camada de rede da pilha de protocolos TCP/IP em sua versão 4 pode ser pensada como um protocolo principal e três protocolos auxiliares. O Protocolo Internet versão 4 (IPv4 – Internet

Protocol version 4), é o principal e é responsável pelas tarefas de empacotamento, encaminhamento e entrega de pacotes na camada de rede. O Protocolo de Mensagens de Controle da Internet versão 4 (ICMPv4 – Internet Control Message Protocol version 4) ajuda o IPv4 a lidar com alguns erros que podem ocorrer na entrega de pacotes na camada de rede. O Protocolo de Gerenciamento de Grupos Internet (IGMP – Internet Group Management Protocol) é usado para ajudar o IPv4 em tarefas de *multicast*. O Protocolo de Resolução de Endereços (ARP – Address Resolution Protocol) é usado para interfacear as camadas de rede e de enlace de dados por meio do mapeamento de endereços da camada de rede em endereços da camada de enlace. A Figura 4.23 mostra as posições desses quatro protocolos na pilha de protocolos TCP/IP.

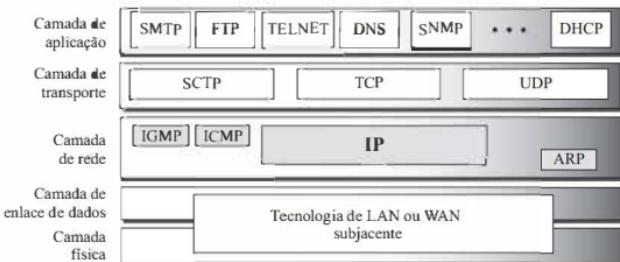



Figura 4.23 Posicionamento do IP e outros protocolos da camada de rede na pilha de protocolos TCP/IP.

Discutiremos o IPv4 e o ICMPv4 nesta seção. O IGMP será discutido quando falarmos sobre protocolos de roteamento. Adiamos a discussão sobre ARP para o Capítulo 5, no qual falamos de endereços da camada de enlace. O IPv4 é um protocolo de datagramas não confiável e não orientado à conexão — um serviço de entrega de melhor esforço. O termo *melhor esforço* significa que os pacotes IPv4 podem ser corrompidos, perdidos, chegar fora de ordem ou atrasados e podem causar congestionamentos na rede. Se a confiabilidade é importante, o IPv4 deve ser acompanhado por um protocolo da camada de transporte que seja confiável, como o TCP. Um exemplo de um serviço de entrega de melhor esforço comumente usado é o serviço dos Correios. Os Correios fazem o possível para entregar cartas normais, mas nem sempre são bem-sucedidos. Se uma carta não registrada for perdida, cabe ao remetente ou ao destinatário-alvo descobrir a perda e corrigir o problema. Os Correios não rastreiam todas as cartas e não são capazes de notificar o remetente sobre sua perda ou dano.

O IPv4 também é um protocolo não orientado à conexão destinado a redes de comutação de pacotes que usam a abordagem de datagramas. Isto significa que cada datagrama é tratado de maneira independente e pode seguir um percurso diferente até seu destino. Isto implica que os datagramas enviados pela mesma origem para o mesmo destino podem chegar fora de ordem. Além disso, alguns deles podem ser perdidos ou danificados durante a transmissão. Novamente, o IPv4 depende de um protocolo de uma camada mais alta para cuidar de todos esses problemas.

Nesta seção, começaremos discutindo o primeiro serviço fornecido pelo IPv4, o empacotamento. Mostraremos como o IPv4 define o formato de um pacote dentro do qual os dados provenientes da camada superior ou de outros protocolos são encapsulados. Em seguida, apresentaremos os endereços IPv4 e, finalmente, discutiremos o roteamento e o encaminhamento de pacotes IPv4 e ICMP.

4.2.1 Formato dos datagramas IPv4

Os pacotes usados pelo IP são denominados ***datagramas***. A Figura 4.24 mostra o formato dos datagramas IPv4. Um datagrama é um pacote de comprimento variável que consiste em duas partes:

cabeçalho e carga útil de dados (*payload*). O cabeçalho tem de 20 a 60 *bytes* de comprimento e contém informações essenciais para o roteamento e a entrega do pacote. No contexto do TCP/IP, é comum mostrar o cabeçalho em seções de 4 *bytes*.

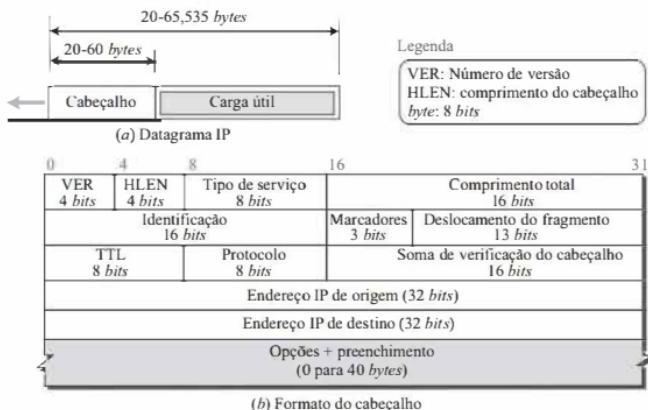



Figura 4.24 Datagrama IP.

Discutir o significado e as razões por trás da existência de cada campo é essencial para entender a operação do IPv4; por isso, a seguir, forneceremos uma breve descrição de cada campo.

- **Número de versão.** O campo de número de versão (VER) de 4 *bits* define a versão do protocolo IPv4, que, obviamente, tem o valor 4.
- **Comprimento do cabeçalho.** O campo de comprimento do cabeçalho (HLEN) de 4 *bits* define o tamanho total do cabeçalho do datagrama em palavras de 4 *bytes*. O datagrama IPv4 tem um cabeçalho de comprimento variável. Quando um dispositivo recebe um datagrama, ele precisa saber quando o cabeçalho acaba e quando os dados, que são encapsulados no pacote, começam. Entretanto, para fazer o valor do comprimento do cabeçalho (número de *bytes*) caber dentro de um campo de 4 *bits*, o comprimento total do cabeçalho é calculado em termos de palavras de 4 *bytes*. O comprimento total, em *bytes*, é dividido por 4 e o valor é inserido neste campo. O receptor tem de multiplicar o valor por 4 para encontrar o comprimento total em *bytes*.
- **Tipo de serviço.** No projeto original do cabeçalho IP, esse campo era conhecido como tipo de serviço (TOS), que definia como o datagrama deveria ser tratado. No final de 1990, a IETF redefiniu esse campo para fornecer *serviços diferenciados* (DiffServ). Quando discutirmos serviços diferenciados no Capítulo 8, ficará mais fácil descrever os *bits* desse campo.
- **Comprimento total.** Esse campo de 16 *bits* define o comprimento total (cabeçalho mais dados), em *bytes*, do datagrama IP. Um número de 16 *bits* pode definir um comprimento total de até 65.535 (quando todos os *bits* são 1s). No entanto, o tamanho do datagrama é normalmente muito menor que isso. Esse campo ajuda o dispositivo receptor a saber quando o pacote chegou por completo.

- **Identificação, marcadores (*flags*) e deslocamento do fragmento.** Esses três campos estão relacionados com a fragmentação do datagrama IP quando o tamanho do datagrama é maior do que a rede subjacente pode transportar. Discutiremos o conteúdo e a importância desses campos quando falarmos sobre fragmentação na próxima seção.
- **Tempo de vida.** Devido ao mau funcionamento de alguns protocolos de roteamento (discutidos mais adiante) um datagrama pode ficar circulando na Internet, visitando algumas redes várias vezes sem chegar ao seu destino. Isso pode criar tráfego desnecessário na Internet. O campo tempo de vida (TTL – Time-To-Live) é usado para controlar o número máximo de saltos (roteadores) que podem ser visitados pelo datagrama. Quando a estação de origem envia o datagrama, ela coloca um número nesse campo. Esse valor é aproximadamente duas vezes o número máximo de saltos (roteadores) entre quaisquer duas estações. Cada roteador, ao processar o datagrama, decrementa esse número de um. Se o valor, após ser decrementado, chegar a zero, o roteador descarta o datagrama.
- **Protocolo.** No TCP/IP, a seção de dados de um pacote, denominada *payload* ou *carga útil*, carrega o pacote completo de outro protocolo. Um datagrama pode, por exemplo, transportar um pacote pertencente a qualquer protocolo da camada de transporte, como UDP ou TCP. Um datagrama pode também carregar um pacote de outros protocolos que utilizam diretamente os serviços do IP, como alguns protocolos de roteamento ou protocolos auxiliares. As autoridades da Internet deram a cada protocolo que utiliza os serviços do IP um número único de 8 *bits*, que é inserido no campo de protocolo. Quando a carga útil é encapsulada em um datagrama IP na origem, o número de protocolo correspondente é inserido nesse campo; quando o datagrama chega ao destino, o valor desse campo ajuda a definir a qual protocolo a carga útil deve ser entregue. Em outras palavras, esse campo permite a multiplexação na origem e demultiplexação no destino, conforme mostra a Figura 4.25. Observe que os campos de protocolo na camada de rede desempenham o mesmo papel que os números de porta na camada de transporte. No entanto, precisamos de dois números de porta em um pacote da camada de transporte porque os números de porta na origem e no destino são diferentes; por outro lado, precisamos de apenas um campo de protocolo porque esse valor é o mesmo para cada protocolo, não importando se ele está sendo utilizado na origem ou no destino.

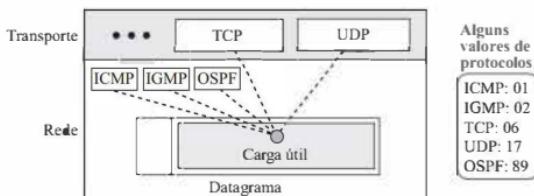



Figura 4.25 Multiplexação e demultiplexação usando o valor do campo de protocolo.

- **Soma de verificação (*checksum*) do cabeçalho.** O IP não é um protocolo confiável. Ele não verifica se a carga útil transportada por um datagrama foi corrompida durante a transmissão. O IP coloca o ônus da verificação de erros da carga útil sobre o protocolo ao qual a carga pertence, tal como o UDP ou o TCP. O cabeçalho do datagrama, no entanto, é adicionado pelo IP e sua verificação de erros é de responsabilidade do IP. Os erros no cabeçalho IP podem ser desastrosos. Por exemplo, se o endereço IP de destino for corrompido, o pacote

pode ser entregue ao *host* errado. Se o campo de protocolo for danificado, a carga útil pode ser entregue ao protocolo errado. Se os campos relacionados com a fragmentação forem corrompidos, o datagrama não pode ser remontado corretamente no destino, e assim por diante. Por essas razões, o IP adiciona um campo de soma de verificação (*checksum*) do cabeçalho exatamente para verificá-lo, mas não a carga útil. É preciso lembrar que, como o valor de alguns campos, como o TTL e outros relacionados com fragmentações e opções, podem mudar de roteador para roteador, o valor da soma de verificação precisa ser recalculado em cada roteador. Conforme foi discutido no Capítulo 3, a soma de verificação na Internet normalmente utiliza um campo de 16 *bits*, que é o complemento da soma dos outros campos calculada utilizando aritmética de complemento de 1s. Discutiremos o cálculo da soma de verificação no Capítulo 5, no qual falaremos sobre detecção de erros.

- **Endereços de origem e de destino.** Esses campos de 32 *bits* definem os endereços IP da origem e do destino, respectivamente. O *host* de origem deve saber seu próprio endereço IP. O endereço IP de destino é conhecido pelo protocolo que usa os serviços do IP ou então é fornecido pelo DNS, conforme descrito no Capítulo 2. Perceba que os valores desses campos devem permanecer inalterados durante todo o tempo que o datagrama IP viaja do *host* de origem ao *host* de destino. Discutiremos mais sobre endereços IP e sua estrutura no final do capítulo.
- **Opções.** O cabeçalho do datagrama pode ter até 40 *bytes* de opções. As opções podem ser usadas para testes de rede e depuração. Embora as opções não sejam uma parte necessária do cabeçalho IP, a capacidade de processamento delas é algo exigido do *software* IP. Isto significa que todas as implementações devem ser capazes de lidar com as opções caso estejam presentes no cabeçalho. A existência de opções em um cabeçalho cria um ônus para o tratamento dos datagramas; algumas opções podem ser alteradas pelo roteadores, o que obriga cada roteador a recalcular o valor da soma de verificação do cabeçalho. Existem opções de um *byte* e de múltiplos *bytes* que discutiremos com mais detalhes no *site* do livro, no material extra do Capítulo 4.
- **Carga útil.** A carga útil (*payload*, ou simplesmente dados), é a principal razão para a criação de um datagrama; é o pacote vindo de outros protocolos que utilizam os serviços do IP. Comparando um datagrama com um pacote postal, a carga útil corresponderia ao conteúdo do pacote; o cabeçalho consistiria apenas na informação escrita na embalagem.

Fragmentação

Um datagrama pode viajar através de redes distintas. Cada roteador desencapsula o datagrama IP do quadro que recebe, o processa e então o encapsula em outro quadro. O formato e o tamanho do quadro recebido dependem do protocolo utilizado pela rede física pela qual o quadro acaba de passar. O formato e o tamanho do quadro enviado dependem do protocolo utilizado pela rede física pela qual o quadro vai passar. Por exemplo, se um roteador conecta uma LAN a uma WAN, ele recebe um quadro no formato da LAN e envia um quadro no formato da WAN.

Unidade Máxima de Transferência

Cada protocolo da camada de enlace (ver Capítulo 5) tem seu próprio formato de quadro. Uma das características de cada formato é o tamanho máximo de carga útil que pode ser encapsulado, conhecido como Unidade Máxima de Transferência (MTU – Maximum Transfer Unit). Em outras palavras, quando um datagrama é encapsulado em um quadro, o tamanho total do datagrama deve ser menor que este tamanho máximo, definido pelas restrições impostas pelo *hardware* e *software* usados na rede (ver Figura 4.26).

O valor da MTU difere de um protocolo da rede física para outro. Por exemplo, o valor para uma LAN é normalmente de 1.500 *bytes*, mas para uma WAN ele pode ser maior ou menor.

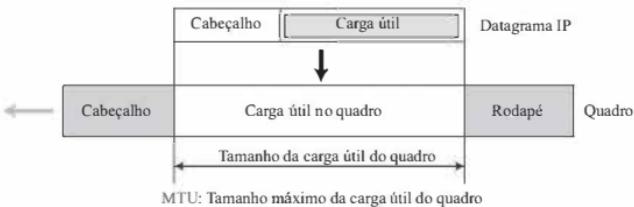



Figura 4.26 Unidade Máxima de Transferência (MTU).

Com o objetivo de tornar o protocolo IP independente da rede física, seus projetistas decidiram fixar o comprimento máximo do datagrama IP em 65.535 *bytes*. Isto garante uma transmissão mais eficiente se um dia usarmos um protocolo da camada de enlace com um MTU desse tamanho. No entanto, para outras redes físicas, temos que dividir o datagrama para permitir que ele passe por tais redes. Isso é chamado **fragmentação**.

Quando um datagrama é fragmentado, cada fragmento tem seu próprio cabeçalho com a maioria dos campos repetidos, mas alguns são alterados. Um datagrama fragmentado pode vir a ser fragmentado caso encontre uma rede com uma MTU ainda menor. Ou seja, um datagrama pode ser fragmentado várias vezes antes de chegar ao seu destino final.

Um datagrama pode ser fragmentado pela estação de origem ou por qualquer roteador no caminho até seu destino. A remontagem do datagrama, no entanto, é feita apenas pela estação de destino, porque cada fragmento se torna um datagrama independente. Considerando que o datagrama fragmentado pode viajar por rotas distintas e que jamais conseguimos controlar ou garantir qual rota seguirá, todos os fragmentos pertencentes ao mesmo datagrama devem, em algum momento, chegar à estação de destino. Por isso, faz sentido deixar a remontagem para o destino final. Uma objeção, ainda mais forte com relação à remontagem dos pacotes durante a transmissão, é a perda de eficiência na qual isto incorreria.

Quando falamos de fragmentação, queremos dizer que a carga útil do datagrama IP é fragmentada. No entanto, a maioria das partes do cabeçalho, exceto por algumas opções, deve ser copiada em todos os fragmentos. A estação ou roteador que fragmenta um datagrama deve alterar os valores de três campos: marcadores (*flags*), deslocamento do fragmento e comprimento total. Os campos restantes devem ser simplesmente copiados. É claro que o valor da soma de verificação deve ser recalculado, independentemente da fragmentação.

Campos relacionados com a fragmentação

Mencionamos anteriormente que existem três campos em um datagrama IP relacionados com a fragmentação: *identificação*, *marcadores* (*flags*) e *deslocamento do fragmento*. Explicaremos esses campos agora.

O campo de identificação de 16 *bits* identifica um datagrama proveniente da estação de origem. A combinação da identificação com o endereço IP de origem deve definir univocamente um datagrama que vem da estação de origem. Para garantir a unicidade, o protocolo IP utiliza um contador para rotular os datagramas, inicializado com um número positivo. Quando o protocolo IP envia um datagrama, ele copia o valor atual do contador para o campo de identificação e incrementa o contador de um. Contanto que o contador seja mantido na memória principal, a unicidade é garantida. Quando um datagrama é fragmentado, o valor no campo de identificação é copiado para todos os fragmentos. Em outras palavras, todos os fragmentos devem ter o mesmo número de identificação, que também é o mesmo do datagrama original, e que ajuda o destino a remontar o datagrama, pois ele sabe que todos os fragmentos que têm o mesmo valor de identificação devem ser remontados como um único datagrama.

O campo de *marcadores* (*flags*) de 3 *bits* define três marcadores. O *bit* mais à esquerda é reservado (não utilizado). O segundo *bit* (*bit* D) é denominado *bit de não fragmentação*. Se o seu valor for 1, o dispositivo não deve fragmentar o datagrama. Se o dispositivo não puder encaminhar o datagrama por meio das redes físicas disponíveis, ele descarta o datagrama e envia uma mensagem ICMP de relatório de erro para a estação de origem (conforme discutido mais adiante). Se o valor do campo for 0, o datagrama pode ser fragmentado, caso seja necessário. O terceiro *bit* (*bit* M) é denominado *bit de mais fragmentos*. Se seu valor for 1, isso significa que o datagrama não é o último fragmento, ou seja, existem mais fragmentos após este. Se o seu valor for 0, isto significa que é o último ou o único fragmento.

O campo *deslocamento do fragmento* de 13 *bits* indica a posição relativa desse fragmento em relação ao datagrama completo. Ele corresponde ao deslocamento dos dados no datagrama original medidos em unidades de 8 *bytes*. A Figura 4.27 mostra um datagrama com um tamanho de dados de 4.000 *bytes* fragmentado em três fragmentos. Os *bytes* do datagrama original são numerados de 0 a 3.999. O primeiro fragmento transporta os *bytes* de 0 a 1.399. O deslocamento para esse datagrama é 0/8 = 0. O segundo fragmento carrega os *bytes* de 1.400 a 2.799, de modo que o valor do deslocamento para este fragmento é 1.400/8 = 175. Finalmente, o terceiro fragmento transporta os *bytes* de 2.800 a 3.999. O valor do deslocamento para este fragmento é 2.800/8 = 350.

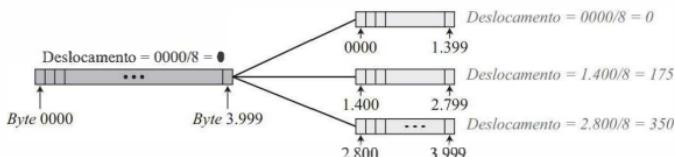



Figura 4.27 Exemplo de fragmentação.

Lembre-se de que o valor do deslocamento é medido em unidades de 8 *bytes*. Isto é feito porque o comprimento do campo de deslocamento é de apenas 13 *bits* e, portanto, não é capaz de representar uma sequência de *bytes* maior que 8.191. Isto força as estações ou roteadores que fragmentam os datagramas a escolherem o tamanho de cada fragmento de modo que o número do primeiro *byte* seja divisível por 8.

A Figura 4.28 mostra uma visão expandida dos fragmentos da figura anterior. O pacote original é enviado pelo cliente, e os fragmentos são remontados no servidor. O valor do campo de identificação é o mesmo em todos os fragmentos. O mesmo vale para o valor do campo de *marcadores*, cujo *bit* de *mais fragmentos* é 1 para todos os fragmentos, exceto o último. A figura também mostra o valor do campo de deslocamento para cada fragmento. Observe que, embora os fragmentos cheguem fora de ordem ao destino, eles podem ser corretamente remontados.

A Figura 4.28 também mostra o que acontece se um fragmento em si é fragmentado. Nesse caso, o valor do campo de deslocamento é sempre relativo ao datagrama original. Por exemplo, conforme mostra a figura, o segundo fragmento é fragmentado mais tarde em dois fragmentos de 800 e 600 *bytes*, mas o deslocamento fornece a posição relativa dos fragmentos em relação os dados originais.

É óbvio que, mesmo se cada fragmento tomar um caminho diferente e chegar na estação de destino fora de ordem, a estação pode remontar o datagrama original a partir dos fragmentos recebidos (caso nenhum deles tenha se perdido), utilizando a seguinte estratégia:

a. O primeiro fragmento tem um campo de deslocamento com valor zero.

b. Dividir o comprimento do primeiro fragmento por 8. O segundo fragmento tem um valor de deslocamento igual ao resultado do cálculo.

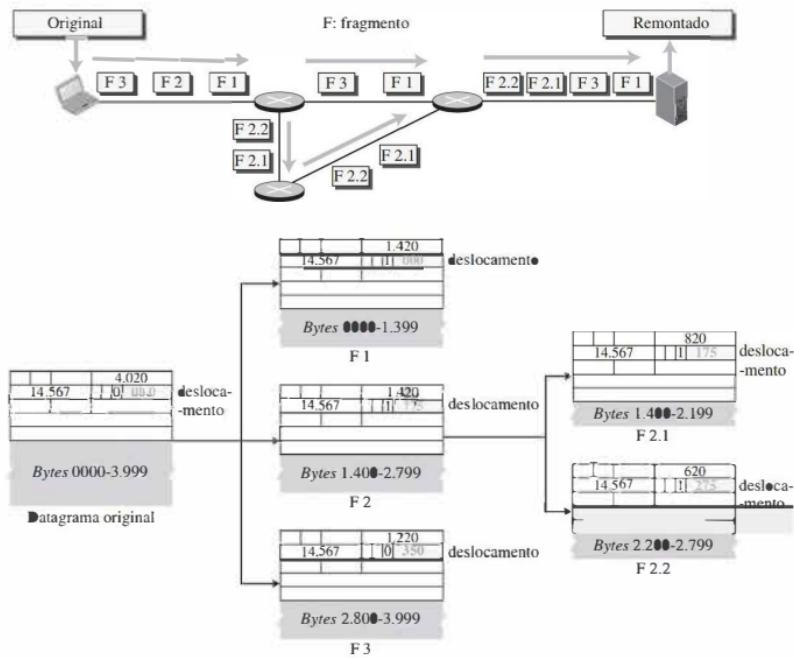



Figura 4.28 Exemplo detalhado de fragmentação.

c. Dividir o comprimento total do primeiro e do segundo fragmentos por 8. O terceiro fragmento tem um valor de deslocamento igual ao resultado do cálculo.

d. Continuar o processo. O último fragmento tem seu *bit* M (*more fragments*) com valor fixado em 0.

Segurança dos datagramas IPv4

O protocolo IPv4, assim como toda a Internet, surgiu quando os usuários da Internet confiavam uns nos outros. Nenhum mecanismo de segurança foi previsto para o protocolo IPv4. Hoje em dia, porém, a situação é diferente; a Internet não é mais segura. Apesar de discutirmos questões de segurança de redes em geral e a segurança do IP em particular no Capítulo 10, a seguir apresentaremos uma breve ideia dos problemas de segurança no protocolo IP e das soluções existentes. Existem três questões de segurança particularmente aplicáveis ao protocolo IP: escuta de pacotes, modificação de pacotes e falsificação de IP.

Escuta de pacotes

Um intruso pode interceptar um pacote IP e fazer uma cópia dele. A escuta de pacotes (*packet sniffing*) é um ataque passivo, no qual o atacante não altera o conteúdo do pacote. Esse tipo de ataque é muito

difícil de ser detectado, porque o emissor e o receptor podem nunca ficar sabendo que o pacote foi copiado. Apesar de não ser possível impedir a escuta de pacotes, a cifração do pacote pode tornar inútil o esforço do atacante. O atacante ainda pode recuperar o pacote, mas seu conteúdo não será inteligível.

Modificação de pacotes

O segundo tipo de ataque é a modificação do pacote. O atacante intercepta o pacote, altera seu conteúdo e envia o novo pacote para o receptor. O receptor acredita que o pacote está vindo do remetente original. Esse tipo de ataque pode ser detectado com o uso de algum mecanismo que forneça integridade aos dados. O receptor, antes de abrir a mensagem e usar seu conteúdo, pode usar esse mecanismo para ter certeza de que o pacote não foi modificado durante a transmissão. Discutiremos mecanismos de integridade no Capítulo 10.

Falsificação de IP (IP spoofing)

Um atacante pode se passar por outra pessoa e criar um pacote IP que carrega o endereço de origem de outro computador. Um atacante pode enviar um pacote IP para um banco fingindo que a origem do pacote é um dos clientes do banco. Esse tipo de ataque pode ser evitado com o uso de mecanismos de autenticação da origem (ver Capítulo 10).

IPSec

Atualmente, os pacotes IP podem ser protegidos contra os ataques mencionados anteriormente usando um protocolo chamado Segurança IP (IPSec – IP Security). Esse protocolo, usado em conjunto com o protocolo IP, cria um serviço orientado à conexão entre duas entidades, permitindo que elas troquem pacotes IP sem se preocupar com os três ataques discutidos anteriormente. Discutiremos o IPSec em detalhes no Capítulo 10; aqui, basta mencionar que o IPSec fornece os seguintes quatro serviços:

- **Definição de algoritmos e chaves.** As duas entidades que querem criar um canal seguro entre elas podem chegar a um acordo em relação a alguns algoritmos disponíveis e às chaves a serem utilizadas para fins de segurança.
- **Cifração de pacotes.** Os pacotes trocados entre as duas entidades podem ser cifrados para fornecer privacidade aos dados, usando um dos algoritmos de criptografia e a chave compartilhada acordada na etapa anterior. Isto torna inúteis os ataques de escuta de pacotes.
- **Integridade dos dados.** A integridade dos dados garante que o pacote não seja modificado durante a transmissão de forma imperceptível. Se o pacote recebido não passar no teste de integridade de dados, é descartado. Isto impede o segundo ataque descrito anteriormente, que consiste na modificação de pacotes.
- **Autenticação da origem.** O IPSec pode autenticar a origem do pacote para garantir que o pacote não tenha sido criado por um impostor. Isto pode evitar ataques de falsificação de IP, conforme foi descrito anteriormente.

4.2.2 Endereços IPv4

O identificador usado na camada IP da pilha de protocolos TCP/IP para identificar a conexão de cada dispositivo à Internet é chamado endereço Internet ou endereço IP. Um endereço IPv4 é um endereço de 32 *bits* que univoca e universalmente define a conexão de uma estação ou de um roteador à Internet. O endereço IP é o endereço da conexão, e não da estação ou do roteador, pois se o dispositivo for movido para outra rede, o endereço IP pode ser alterado.

Os endereços IPv4 são únicos no sentido de que cada endereço define uma, e somente uma, conexão com a Internet. Se um dispositivo tem duas conexões com a Internet, por meio de duas redes,

ele tem dois endereços IPv4. Os endereços IPv4 são universais, já que o sistema de endereçamento deve ser aceito por qualquer estação que queira ser conectada à Internet.

Espaço de endereços

Um protocolo como o IPv4, que define endereços, apresenta um espaço de endereços. Um espaço de endereços corresponde ao número total de endereços usados pelo protocolo. Se um protocolo utiliza *b bits* para definir um endereço, o espaço de endereços é 2^b porque cada *bit* pode ter dois valores distintos (0 ou 1). O IPv4 utiliza endereços de 32 *bits*, o que significa que o espaço de endereçamento é 2^{32} ou 4.294.967.296 (mais do que quatro bilhões). Se não houvesse restrições, mais de quatro bilhões de dispositivos poderiam ser conectados à Internet.

Notação

Há três notações comuns para representar um endereço IPv4: a notação binária (base 2), a notação decimal com pontos (base 256) e a notação hexadecimal (base 16). Na *notação binária*, um endereço IPv4 é representado na forma de 32 *bits*. Para tornar o endereço mais legível, um ou mais espaços são geralmente inseridos entre cada octeto (8 *bits*). Cada octeto é normalmente denominado *byte*. Para tornar o endereço IPv4 mais compacto e mais fácil de ler, ele é normalmente escrito no formato decimal, com um ponto separando os *bytes*. Esse formato é chamado notação *decimal com pontos*. Observe que, como cada *byte* (octeto) tem apenas 8 *bits*, cada número na notação *decimal com pontos* possui um valor entre 0 e 255. Algumas vezes, encontramos um endereço IPv4 na notação hexadecimal. Cada dígito hexadecimal equivale a quatro *bits*. Isto significa que um endereço de 32 *bits* apresenta 8 dígitos hexadecimais. Essa notação é frequentemente utilizada na programação para a rede. A Figura 4.29 mostra um endereço IP que usa as três notações discutidas.

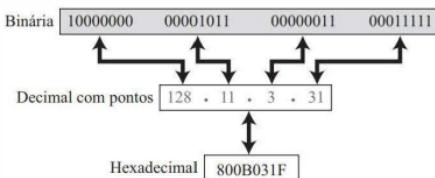



Figura 4.29 Três notações diferentes usadas no endereçamento IPv4.

Hierarquia no endereçamento

Em qualquer rede de comunicação que envolve entregas, como em uma rede de telefonia ou dos correios, o sistema de endereçamentos é hierárquico. Na rede dos Correios, o endereço postal inclui o país, estado, cidade, rua, número e o nome do destinatário da mensagem. De forma similar, um número de telefone é dividido em código do país, código de área, número local e ramal.

Um endereço IPv4 de 32 *bits* também é hierárquico, mas é dividido em apenas duas partes. A primeira parte do endereço, chamada *prefixo*, define a rede; a segunda parte do endereço, chamada *sufixo*, define o nó (a conexão de um dispositivo à Internet). A Figura 4.30 mostra o prefixo e o sufixo de um endereço IPv4 de 32 *bits*. O comprimento do prefixo é *n bits* e o comprimento do sufixo é (32 – *n*) *bits*.

Um prefixo pode ter um comprimento fixo ou variável. O identificador de rede no IPv4 foi inicialmente projetado para usar prefixos de comprimento fixos. Esse esquema, que atualmente está obsoleto, é conhecido como endereçamento com classes (*classful*). O novo esquema, conhecido

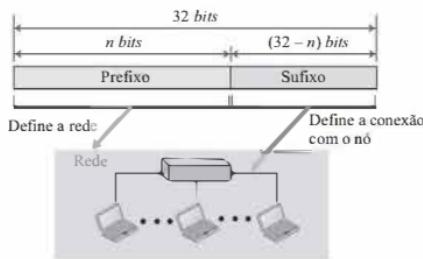



Figura 4.30 Hierarquia no endereçamento.

como endereçamento sem classes (*classless*), utiliza prefixos de rede de comprimento variável. Primeiramente, discutimos brevemente o endereçamento com classes; em seguida, abordaremos o endereçamento sem classes.

Endereçamento com classes

Nos primórdios da Internet, o endereçamento IPv4 foi projetado com um prefixo de comprimento fixo. Entretanto, para acomodar tanto redes pequenas como grandes, três prefixos foram criados em vez de apenas um ($n = 8$, $n = 16$ e $n = 24$). O espaço de endereços completo foi dividido em cinco classes (classe A, B, C, D e E), conforme mostra a Figura 4.31. Esse esquema é denominado **endereçamento com classes** (*classful addressing*).

Classe	Prefixos	Primeiro *byte*
A	$n = 8$ *bits*	0 a 127
B	$n = 16$ *bits*	128 a 191
C	$n = 24$ *bits*	192 a 223
D	Não se aplica	224 a 239
E	Não se aplica	240 a 255

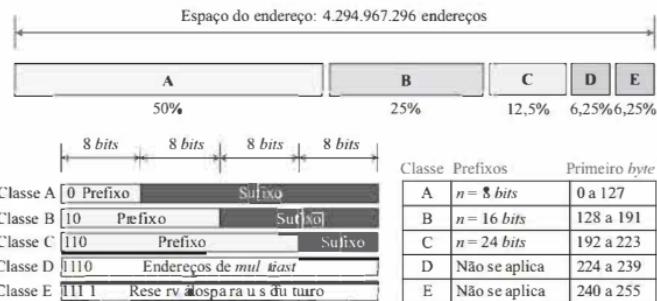



Figura 4.31 Ocupação do espaço de endereços no endereçamento com classes.

Na classe A, o comprimento da rede é de 8 *bits*, mas como o primeiro *bit*, que é 0, define a classe, podemos usar apenas sete *bits* para o identificador da rede. Isto significa que existem apenas $2^7 = 128$ redes no mundo que podem ter um endereço de classe A.

Na classe B, o comprimento da rede é de 8 *bits*, mas como os dois primeiros *bits*, cujo valor é $(10)_2$, definem a classe, podemos usar apenas 14 *bits* para o identificador da rede. Isto significa que existem apenas $2^{14} = 16.384$ redes no mundo que podem ter um endereço de classe B.

Todos os endereços que começam com $(110)_2$ pertencem à classe C. Nela, o comprimento da rede é de 24 *bits*, mas como três destes *bits* definem a classe, podemos usar apenas 21 *bits* para o identificador da rede. Isto significa que existem 2^{21} = 2.097.152 redes no mundo que podem ter um endereço de classe C.

A classe D não é dividida em prefixo e sufixo, pois é usada para endereços de *multicast*. Todos os endereços que começam com 1111 em binário pertencem à classe E. Como na classe D, a classe E não é dividida em prefixo e sufixo; é reservada para uso futuro.

Esgotamento de endereços

A razão pela qual o endereçamento com classes tornou-se obsoleto é o esgotamento dos endereços. Como os endereços não foram distribuídos de forma adequada, a Internet passou a sofrer com o problema dos endereços estarem sendo rapidamente utilizados, resultando na falta de endereços disponíveis para organizações e indivíduos que precisavam ser conectados à Internet. Para entender o problema, considere a classe A. Ela pode ser atribuída a apenas 128 organizações em todo o mundo, mas cada organização precisa ter uma única rede (vista pelo resto do mundo) com 16.777.216 nós (computadores na mesma rede). Como provavelmente existe um número pequeno de organizações tão grandes, a maioria dos endereços nessa classe acabou desperdiçada (não utilizada). Os endereços da classe B foram projetados para organizações de médio porte, mas muitos dos endereços dessa classe também acabaram não sendo utilizados. Os endereços de classe C têm uma falha de projeto completamente diferente. O número de endereços que podem ser usados em cada rede (256) era tão pequeno que a maioria das empresas não se sentia confortável em usar um bloco de endereços desta classe. Os endereços da classe E quase nunca foram usados, levando a um desperdício da classe toda.

Sub-redes e super-redes

Para aliviar o esgotamento de endereços, duas estratégias foram propostas e, até certo ponto, implementadas: a criação de sub-redes e de super-redes. Para criar uma sub-rede, um bloco da classe A ou B é dividido em vários blocos menores. Cada sub-rede tem um comprimento maior que o prefixo da rede original. Por exemplo, se uma rede da classe A é dividida em quatro sub-redes, cada sub-rede recebe um prefixo igual a $n_{sub} = 10$. Ao mesmo tempo, caso nem todos os endereços de uma rede sejam utilizados, a criação de sub-redes permite que o espaço de endereços seja dividido entre diversas organizações. Essa ideia não deu certo, porque a maioria das grandes organizações não ficou feliz em dividir seus blocos e dar alguns dos endereços não utilizados para organizações menores.

Enquanto as sub-redes foram criadas para dividir um grande bloco em partes menores, as super-redes foram concebidas para combinar diversos blocos da classe C em um bloco maior, que se tornaria mais atraente para organizações que precisavam de mais do que os 256 endereços disponíveis em um bloco da classe C. Essa ideia também não funcionou porque dificultava o roteamento de pacotes.

Vantagens do endereçamento com classes

Embora o endereçamento com classes tenha vários problemas e tenha se tornado obsoleto, ele apresentava uma vantagem: dado um endereço, podemos encontrar facilmente a classe do endereço e, como o comprimento do prefixo para cada classe é fixo, somos capazes de descobri-lo imediatamente. Em outras palavras, o comprimento do prefixo no tratamento com classes é inerente ao endereço; nenhuma informação adicional é necessária para extrair o prefixo e o sufixo.

Endereçamento sem classes

A criação de sub-redes e super-redes no endereçamento com classes não resolvia de fato o problema de esgotamento de endereços. Com o crescimento da Internet, ficou claro que um espaço de endereços maior era necessário como uma solução de longo prazo. O maior espaço de endereços exigia, entretanto,

que o tamanho dos endereços IP também fosse aumentado, ou seja, o formato dos pacotes IP precisaria ser alterado. Embora a solução de longo prazo já tenha sido concebida e seja denominada IPv6 (discutido mais adiante), uma solução de curto prazo também foi projetada para utilizar o mesmo espaço de endereços, mas alterando a distribuição de endereços para fornecer a cada organização um bloco de endereços de tamanho adequado. A solução de curto prazo ainda usa endereços IPv4, mas é chamada **endereçamento sem classes** (*classless addressing*). Ou seja, a existência das classes foi removida da distribuição dos endereços para compensar o seu esgotamento.

Havia uma outra motivação para o endereçamento sem classes. Durante a década de 1990, os Provedores de Serviços de Internet (ISPs – Internet Service Providers) passaram a ganhar destaque. Um ISP é uma organização que fornece acesso à Internet para indivíduos, pequenas empresas e organizações de médio porte que não querem controlar uma parte da Internet e se envolver na prestação de serviços da Internet (por exemplo, correio eletrônico) para seus funcionários. Um ISP pode fornecer esses serviços. Um ISP recebe um grande número de endereços e depois os subdivide em grupos de 1, 2, 4, 8, 16 e assim por diante, fornecendo uma faixa de endereços para uma residência ou para uma empresa de pequeno porte. Os clientes são conectados ao ISP via conexão discada (por exemplo: DSL) ou via cabo. Entretanto, cada cliente precisa de alguns endereços IPv4.

Em 1996, as autoridades da Internet anunciaram a nova arquitetura de endereçamento sem classes; nele, são usados blocos de tamanhos variáveis que não pertencem a qualquer classe. Podemos ter um bloco de 1 endereço, 2 endereços, 4 endereços, 128 endereços, e assim por diante.

No endereçamento sem classes, o espaço de endereços completo é dividido em blocos de comprimento variável. O prefixo de um endereço define o bloco (a rede); o sufixo determina o nó (o dispositivo). Teoricamente, podemos ter um bloco de 2^0, 2^1, 2^2, ..., 2^{32} endereços. No entanto, uma das restrições, conforme discutido mais adiante, é que o número de endereços em um bloco deve ser uma potência de 2. Uma organização pode receber um bloco de endereços. A Figura 4.32 mostra a divisão do espaço de endereços completo em blocos não sobrepostos.

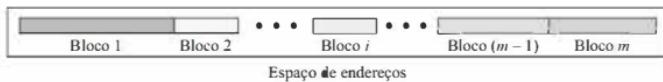



Figura 4.32 Blocos de tamanho variável no endereçamento sem classes.

Diferentemente do endereçamento com classes, o comprimento do prefixo no endereçamento sem classes é variável. Podemos ter um comprimento de prefixo que varia de 0 a 32. O tamanho da rede é inversamente proporcional ao comprimento do prefixo. Um prefixo pequeno significa uma rede maior; um prefixo grande, uma rede menor.

É preciso enfatizar que a ideia de endereçamento sem classes pode ser facilmente aplicada ao endereçamento com classes. Um endereço da classe A pode ser pensado como um endereço sem classes cujo comprimento do prefixo é 8. Um endereço da classe B pode ser pensado como um endereço sem classes no qual o prefixo é 16, e assim por diante. Em outras palavras, o endereçamento com classes é um caso particular do endereçamento sem classes.

Comprimento do prefixo: Notação de barra

A primeira pergunta que precisamos responder ao lidar com endereçamento sem classes é como encontrar o comprimento do prefixo dado um endereço. Como o comprimento do prefixo não é inerente ao endereço, precisamos fornecer o comprimento do prefixo separadamente. Nesse caso, o comprimento do prefixo, *n*, é adicionado ao endereço, separado por uma barra. A notação é informalmente conhecida como *notação de barra* e formalmente como estratégia **Roteamento Interdomínios sem Classes** (CIDR – Classless Interdomain Routing). Um endereço no endereçamento sem classes pode então ser representado conforme mostrado na Figura 4.33.

Figura 4.33 Notação de barra (CIDR).

Em outras palavras, um endereço no endereçamento sem classes não define, por si só, o bloco ou a rede à qual o endereço pertence; precisamos também fornecer o comprimento do prefixo.

Extraindo informações de um endereço

Dado qualquer endereço no bloco de endereços, normalmente precisamos determinar três informações sobre o bloco ao qual o endereço pertence: o número total de endereços, o primeiro endereço do bloco e o último dos endereços. Como o valor do comprimento do prefixo, n, é fornecido, pode-se encontrar facilmente essas três informações, conforme mostra a Figura 4.34.

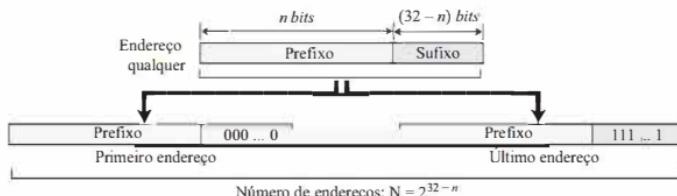



Figura 4.34 Extraindo informações no endereçamento sem classes.

1. O número de endereços no bloco é $N = 2^{32-n}$.
2. Para determinar o primeiro endereço, tomamos os n *bits* mais à esquerda e fixamos os $(32 - n)$ *bits* mais à direita em 0.
3. Para determinar o último endereço, tomamos os n *bits* mais à esquerda e fixamos os $(32 - n)$ *bits* mais à direita em 1.

Exemplo 4.1

Um endereço sem classes é dado por 167.199.170.82/**27**. Podemos determinar as três informações anteriores usando o procedimento a seguir. O número de endereços na rede é $2^{32-n} = 2^5 = 32$ endereços.

O primeiro endereço pode ser determinado mantendo-se os primeiros 27 *bits* inalterados e fixando o restante dos *bits* em 0.

Endereço: 167.199.170.82/**27**	**10100111 11000111 10101010 01010010**
Primeiro endereço: 167.199.170.64/**27**	**10100111 11000111 10101010 01000000**

O último endereço pode ser determinado mantendo-se os primeiros 27 *bits* inalterados e fixando o restante dos *bits* em 1.

Endereço: 167.199.170.82/**27**	**10100111 11000111 10101010 01011111**
Último endereço: 167.199.170.95/**27**	**10100111 11000111 10101010 01011111**

Máscara de endereço

Outra maneira de obter o primeiro e o último endereços do bloco é usar a máscara do endereço, que é um número de 32 *bits* no qual os *n* *bits* mais à esquerda são fixados em 1 e o restante dos *bits* (32 – *n*) são fixados em 0. Um computador pode facilmente encontrar a máscara de endereço porque ela é o complemento de $(2^{32-n} - 1)$. A razão pela qual se define uma máscara dessa maneira é que ela pode ser usada por um programa de computador para extrair as informações de um bloco, aplicando as operações binárias NOT (inversão), AND ('e' *bit* a *bit*) e OR ('ou' *bit* a *bit*).

1. O número de endereços do bloco N = NOT (Máscara) + 1.
2. O primeiro endereço do bloco = (Qualquer endereço do bloco) AND (Máscara).
3. O último endereço do bloco = (Qualquer endereço do bloco) OR [NOT (Máscara)].

Exemplo 4.2

Vamos repetir o Exemplo 4.1 usando a máscara. A máscara em notação decimal com pontos é 256.256.256.224. As operações AND, OR e NOT podem ser aplicadas a *bytes* individuais usando calculadoras e *applets* no *site* do livro.

Número de endereços do bloco:	N = **NOT** (máscara) + 1= 0.0.0.31 + 1 = 32 endereços
Primeiro endereço:	Primeiro = (endereço) **AND** (máscara) = 167.199.170. 82
Último endereço:	Último = (endereço) **OR** (**NOT** máscara) = 167.199.170. 255

Exemplo 4.3

No endereçamento sem classes, um endereço não é capaz de definir, por si só, o bloco de endereços ao qual ele pertence. Por exemplo, o endereço 230.8.24.56 pode pertencer a muitos blocos. Alguns deles são mostrados a seguir com o valor do prefixo associado a esse bloco.

Comprimento do prefixo:16	→	Bloco: 230.8.0.0	a	230.8.255.255
Comprimento do prefixo:20	→	Bloco: 230.8.16.0	a	230.8.31.255
Comprimento do prefixo:26	→	Bloco: 230.8.24.0	a	230.8.24.63
Comprimento do prefixo:27	→	Bloco: 230.8.24.32	a	230.8.24.63
Comprimento do prefixo:29	→	Bloco: 230.8.24.56	a	230.8.24.63
Comprimento do prefixo:31	→	Bloco: 230.8.24.56	a	230.8.24.57

Endereço de rede

Os exemplos anteriores mostram que, dado qualquer endereço, podemos determinar todas as informações sobre o seu bloco. O primeiro endereço, o **endereço de rede**, é particularmente importante porque é usado no roteamento de um pacote até sua rede de destino. Por ora, considere uma internet composta por *m* redes e um roteador com *m* interfaces. Quando um pacote chega ao roteador vindo de qualquer estação de origem, o roteador precisa saber para qual rede o pacote deve ser encaminhado: por qual interface o pacote deve sair. Quando o pacote chega a esta rede, chega à sua estação de destino usando uma outra estratégia que discutiremos mais adiante. A Figura 4.35 ilustra a ideia. Depois de determinar o endereço de rede, o roteador consulta sua tabela de roteamento para determinar a interface correspondente pela qual o pacote deve ser enviado. O endereço de rede corresponde, na verdade, ao identificador da rede; cada rede é identificada por seu endereço de rede.

Alocação de blocos

Outra questão no endereçamento sem classes refere-se à alocação de blocos. Como os blocos são alocados? A responsabilidade final sobre a alocação de blocos é atribuída a uma autoridade mundial

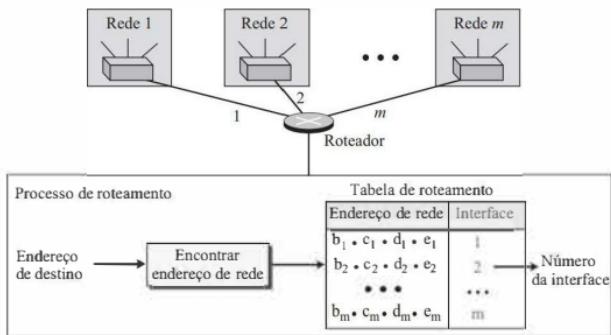



Figura 4.35 Endereço de rede.

denominada Corporação para Atribuição de Nomes e Números na Internet (ICANN – Internet Corporation for Assigned Names and Numbers). No entanto, a ICANN normalmente não aloca endereços para usuários individuais da Internet. Ela atribui um grande bloco de endereços a um ISP (ou uma grande organização que é considerada um ISP nesse caso). Para a correta operação do CIDR, duas restrições precisam ser aplicadas ao bloco alocado.

1. O número de endereços solicitados, denotado N, deve ser uma potência de 2. A razão para isso é que $N = 2^{32-n}$ ou $n = 32 - \log_2 N$. Se N não for uma potência de 2, não é possível obter um valor inteiro para n.
2. O bloco solicitado precisa ser alocado onde exista um número contíguo de endereços disponíveis no espaço de endereços. Entretanto, existe uma restrição na escolha do primeiro endereço do bloco. O primeiro endereço precisa ser divisível pelo número de endereços do bloco. A razão é que o primeiro endereço deve ser o prefixo seguido por um número de 0s igual a $(32 - n)$. O valor decimal do primeiro endereço é, portanto,

primeiro endereço = (prefixo em decimal) × 2^{32-n} = (prefixo em decimal) × N.

Exemplo 4.4

Um ISP solicitou um bloco contendo 1.000 endereços. Como 1.000 não é uma potência de 2, são concedidos 1.024 endereços a esse provedor. O comprimento do prefixo é calculado como $n = 32 - \log_2 1.024 = 22$. Um bloco disponível, 18.14.12.0/**22**, é concedido ao ISP. Pode-se notar que o primeiro endereço em decimal é 302.910.464, divisível por 1.024.

Sub-redes

Pode-se criar mais níveis na hierarquia por meio de sub-redes. Uma organização (ou um ISP) que recebe uma faixa de endereços pode dividi-la em diversas subfaixas e atribuir cada subfaixa a uma sub-rede. Perceba que nada impede a organização de criar mais níveis. Uma sub-rede pode ser dividida em várias subsub-redes. Uma subsub-rede pode ser dividida em várias subsubsub-redes, e assim por diante.

Projetando sub-redes As sub-redes em uma rede devem ser cuidadosamente projetadas para permitir o roteamento de pacotes. Considere que o número total de endereços concedidos à organização seja N, que o comprimento do prefixo seja n, que o número de endereços atribuído a cada sub-rede seja N_{sub} e que o comprimento do prefixo de cada sub-rede seja n_{sub}. Nesse contexto, os passos a seguir devem ser seguidos cuidadosamente para garantir a correta operação das sub-redes.

- O número de endereços em cada sub-rede deve ser uma potência de 2.
- O comprimento do prefixo para cada sub-rede deve ser calculado por meio da seguinte fórmula:

$$n_{sub} = 32 - \log_2 N_{sub}$$

- O endereço inicial de cada sub-rede deve ser divisível pelo número de endereços daquela sub-rede. Pode-se conseguir isto se começarmos a atribuição de endereços pelas sub-redes maiores.

Determinando informações sobre cada sub-rede Depois de projetar as sub-redes, as informações sobre cada sub-rede, como o primeiro e último endereços, podem ser determinadas usando o processo que descrevemos para encontrar as informações sobre cada rede na Internet.

Exemplo 4.5

Uma organização recebe um bloco de endereços cujo endereço inicial é 14.24.74.0/24. A organização precisa ter três sub-blocos de endereços para usar em suas três sub-redes: um sub-bloco de 10 endereços, um sub-bloco de 60 endereços e um sub-bloco de 120 endereços. Projete os sub-blocos.

Solução

Existem $2^{32-24} = 256$ endereços nesse bloco. O primeiro endereço é 14.24.74.0/24; o último endereço é 14.24.74.255/24. Para satisfazer o terceiro requisito, vamos atribuir endereços aos sub-blocos, começando pelo maior e terminando com o menor.

a. O número de endereços no maior sub-bloco, que requer 120 endereços, não é uma potência de 2. Alocamos, portanto, 128 endereços a essa sub-rede. A máscara de sub-rede correspondente pode ser calculada como $n_1 = 32 - \log_2 128 = 25$. O primeiro endereço nesse bloco é 14.24.74.0/25 e o último é 14.24.74.127/25.

b. O número de endereços no segundo maior sub-bloco, que requer 60 endereços, também não é uma potência de 2. Alocamos, portanto, 64 endereços a essa sub-rede. A máscara de sub-rede correspondente pode ser calculada como $n_2 = 32 \log_2 64 = 26$. O primeiro endereço nesse bloco é 14.24.74.128/26, enquanto o último é 14.24.74.191/26.

c. O número de endereços no menor sub-bloco, que requer 10 endereços, também não é uma potência de 2. Alocamos, portanto, 16 endereços a essa sub-rede. A máscara de sub-rede correspondente pode ser calculada como $n_3 = 32 \log_2 16 = 28$. O primeiro endereço nesse bloco é 14.24.74.192/28, enquanto o último é 14.24.74.207/28.

Se somarmos todos os endereços nos sub-blocos anteriores, o resultado são 208 endereços, o que significa que 48 endereços são deixados em reserva. O primeiro endereço nessa faixa é 14.24.74.208. O último endereço é 14.24.74.255. Ainda não conhecemos o comprimento de prefixo. A Figura 4.36 mostra a configuração dos blocos, com o primeiro endereço de cada um.

Agregação de endereços

Uma das vantagens da estratégia CIDR é a **agregação de endereços** (também denominada *resumo de endereços* ou *resumo de rotas*). Quando os blocos de endereços são combinados para criar um bloco maior, o roteamento pode ser realizado com base no prefixo do maior bloco. A ICANN atribui grandes blocos de endereços para cada ISP. Cada ISP, por sua vez, divide o bloco a ele atribuído em sub-blocos menores e distribui os sub-blocos a seus clientes.

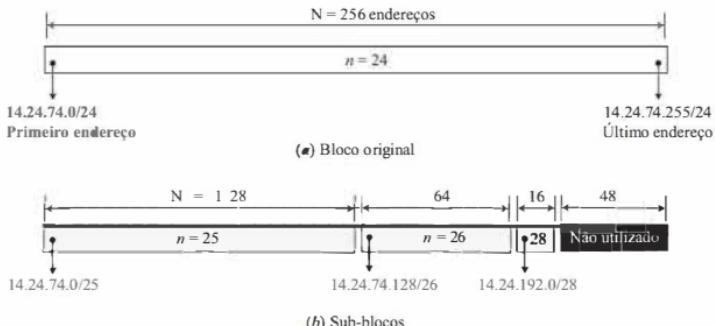



Figura 4.36 Solução do Exemplo 4.5.

Exemplo 4.6

A Figura 4.37 mostra como quatro pequenos blocos de endereços podem ser distribuídos por um ISP a quatro organizações. O ISP combina esses quatro blocos em um só e divulga o maior bloco para o restante do mundo. Qualquer pacote destinado a esse bloco maior deve ser enviado para o ISP, que é responsável por encaminhar o pacote à organização adequada. Isto é semelhante ao roteamento encontrado em redes postais. Todos os pacotes vindos de fora de um país são enviados primeiramente à capital e depois distribuídos para o destino correspondente.

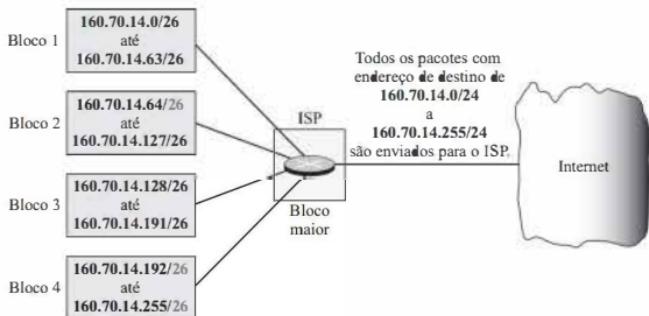



Figura 4.37 Exemplo de agregação de endereços.

Endereços especiais

Antes de finalizar a discussão sobre endereços no IPv4, é preciso mencionar a existência de cinco endereços especiais que são utilizados para propósitos especiais: endereço de *esta estação* (*this-host*), de *broadcast limitado*, de *loopback*, endereços *privados* e de *multicast*.

Endereço de esta estação O único endereço no bloco **0.0.0.0/32** é denominado endereço de *esta estação* (*this-host*). Ele é usado sempre que uma estação precisa enviar um datagrama IP, mas não sabe seu próprio endereço para usar como endereço de origem. Veremos um exemplo dessa situação na próxima seção.

Endereço de *broadcast* limitado O único endereço no bloco **255.255.255.255/32** é denominado endereço de *broadcast* limitado. Ele é usado sempre que um roteador ou uma estação cliente precisa enviar um datagrama para todos os dispositivos em uma rede. Os roteadores da rede, no entanto, bloqueiam os pacotes que têm esse endereço como destino, de modo que o pacote não saia da rede.

Endereço de *loopback* O bloco **127.0.0.0/8** é denominado endereço de *loopback*. Um pacote cujo endereço de destino pertence a esse bloco jamais deixa a interface de rede da estação, ou seja, ele permanece na estação. Qualquer endereço do bloco pode ser usado para testar algum programa sendo executado na máquina. Por exemplo, pode-se escrever um programa-cliente e um programa-servidor de modo que o endereço do servidor seja fixado como um dos endereços deste bloco. Antes de executar estes programas em computadores diferentes, podemos testá-los usando a mesma estação para verificar se eles funcionam.

Endereços privados Existem quatro blocos designados como endereços privados: 10.0.0.0/**8**, 172.16.0.0/**12**, 192.168.0.0/**16** e 169.254.0.0/**16**. Veremos a aplicação desses endereços quando discutirmos o NAT mais adiante neste capítulo.

Endereços de *multicast* O bloco 224.0.0.0/**4** é reservado para endereços de *multicast*. Discutiremos esses endereços mais adiante neste capítulo.

Protocolo de Configuração Dinâmica de *Host*

Vimos que uma grande organização ou um ISP podem receber um bloco de endereços diretamente da ICANN, enquanto uma pequena organização pode recebê-lo de um ISP. Após um bloco de endereços ser atribuído a uma organização, a administração da rede pode atribuir endereços manualmente para estações individuais ou roteadores. Por outro lado, a atribuição de endereços em uma organização também pode ser feita automaticamente usando o **Protocolo de Configuração Dinâmica de *Host*** (DHCP – Dynamic Host Configuration Protocol). O DHCP é um programa da camada de aplicação que, utilizando o paradigma cliente-servidor, auxilia o TCP/IP na camada de rede.

O DHCP experimenta um uso tão difundido na Internet que muitas vezes é chamado *protocolo plug-and-play**. Pode ser usado em diversas situações. Um gerente de rede pode configurar o DHCP para atribuir endereços IP permanentes a estações e roteadores. O DHCP pode também ser configurado para fornecer endereços IP temporários, sob demanda, às estações. Essa segunda funcionalidade permite que se atribua um endereço IP temporário a um usuário que conecta seu *notebook* à Internet enquanto ele estiver hospedado em um hotel e também permite que um ISP que tenha recebido apenas 1.000 endereços preste serviços a 4.000 residências, assumindo que nunca haja mais que um quarto dos clientes usando a Internet ao mesmo tempo.

Além de seu endereço IP, um computador também precisa saber o prefixo da rede (ou máscara de endereço). A maioria dos computadores também precisa de duas outras informações: o endereço do roteador-padrão (*default router*), para que a estação seja capaz de se comunicar com outras redes, e o endereço de um servidor de nomes, para que a estação seja capaz de usar nomes em vez de endereços, conforme discutido no Capítulo 2. Em outras palavras, quatro informações são normalmente necessárias: o endereço do computador, o prefixo da rede, o endereço de um roteador e o endereço IP de um servidor de nomes. O DHCP pode ser usado para fornecer todas essas informações para o *host*.

Formato das mensagens DHCP

O DHCP é um protocolo cliente-servidor no qual o cliente envia uma mensagem de solicitação e o servidor retoma uma mensagem de resposta. Antes de discutirmos o funcionamento do DHCP, mostraremos o formato geral da mensagem DHCP, exibido na Figura 4.38. A maioria dos campos é explicada na figura, mas precisamos discutir o campo de opções, que desempenha um papel muito importante no DHCP.

* N. de T.: O termo "*plug-and-play*" é comumente utilizado para designar tecnologias nas quais basta conectar o dispositivo que ele se configura automaticamente e já fica disponível para utilização.

0 8	16	24	31
Opcode	TipoH	CompH	CountS
ID da transação			
Tempo decorrido		Marcadores	
Endereço IP do cliente			
Seu endereço IP			
Endereço IP do servidor			
Endereço IP do roteador-padrão			
Endereço *hardware* do cliente			
Nome do servidor			
Nome do arquivo de inicialização			
Opções			

Campos:
Opcode: Código de operação, pedido (1) ou resposta (2)
TipoH: Tipo de *hardware* (Ethernet, ...)
CompH: Comprimento do endereço de *hardware*
CountS: Número máximo de saltos pelos quais o pacote passa
ID da transação: Um inteiro fixado pelo cliente e repelido pelo servidor
Tempo decorrido: O número de segundos desde que o cliente começou a inicialização
Marcadores (*Flags*): Primeiro *bit* define *unicast* (0) ou *multicast* (1); os outros 15 *bits* não são usados
Endereço IP do cliente: Fixado em 0 se o cliente não souber seu valor
Seu endereço IP: O endereço IP do cliente enviado pelo servidor
Endereço IP do servidor: Um endereço IP de *broadcast* se o cliente não souber o seu valor
Endereço IP do roteador-padrão: O endereço do roteador-padrão (*default router*, ou *gateway*)
Nome do servidor: O nome de domínio de 64 *bytes* referente ao servidor
Nome do arquivo de inicialização: Um nome de arquivo de 128 *bytes* contendo informações adicionais
Opções: Um campo de 64 *bytes* com dupla finalidade, conforme descrito no texto

Figura 4.38 Formato das mensagens DHCP.

Ele pode transportar qualquer informação adicional ou alguma informação de um fornecedor específico. O servidor usa um número, chamado ***cookie* mágico** (*magic cookie*), no formato de um endereço IP contendo o valor 99.130.83.99. Quando o cliente termina de ler a mensagem, ele procura pelo *cookie* mágico. Se ele estiver presente, os próximos 60 *bytes* são opções. Uma opção é composta por três campos: campo de identificação de 1 *byte*, campo de comprimento de 1 *byte* e campo de valor de comprimento variável. Existem vários campos de identificação que são usados principalmente por fornecedores de equipamentos. Se o campo de identificação for 53, o campo de valor define um dos 8 tipos de mensagens mostrados na Figura 4.39. Mostraremos como esses tipos de mensagens são usados pelo DHCP.

Figura 4.39 Formato das opções.

Operação do DHCP

A Figura 4.40 mostra um cenário simples ilustrativo.

1. A estação que está ingressando na rede cria uma mensagem de **DHCP**DISCOVER na qual apenas o campo ID de transação é preenchido com um número aleatório. Nenhum outro campo pode ser definido porque a estação não tem o conhecimento necessário para fazê-lo. Essa mensagem é encapsulada em um datagrama de usuário UDP com a porta de origem fixada em 68 e com a porta de destino fixada em 67. Discutiremos a razão para usar dois números de porta bem conhecidos mais tarde. O datagrama de usuário é encapsulado em um datagrama IP com o endereço de origem fixado em **0.0.0.0** ("esta estação") e com o endereço de destino fixado em **255.255.255.255** (endereço de *broadcast*). A razão é que a estação ingressante não conhece nem seu próprio endereço nem o endereço do servidor.

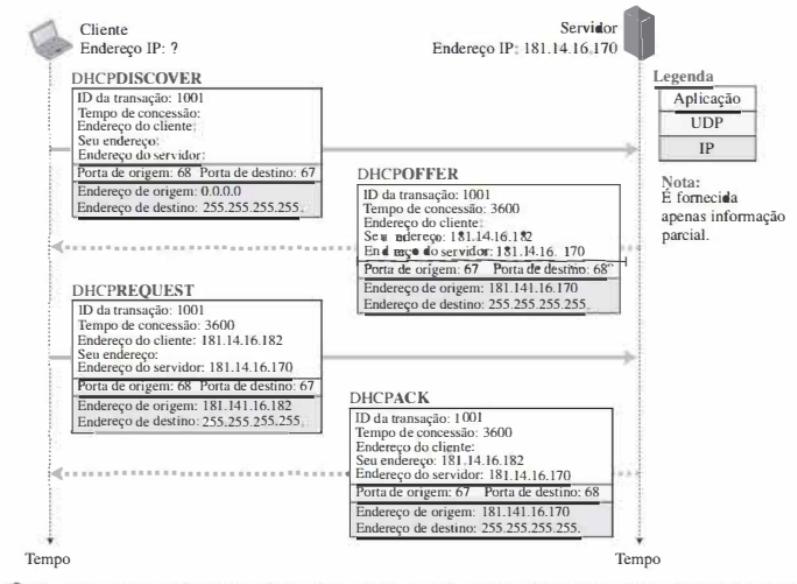



Figura 4.40 Operação do DHCP.

2. O servidor DHCP (ou servidores, se houver mais de um) responde com uma mensagem **DHCP**OFFER na qual o campo "Seu endereço IP" define o endereço IP oferecido para a estação ingressante e o campo "Endereço IP do servidor" inclui o endereço IP do servidor. A mensagem também inclui o tempo de concessão pelo qual o *host* pode usar esse endereço IP. Essa mensagem é encapsulada em um datagrama de usuário com os mesmos números de porta, mas na ordem inversa. O datagrama de usuário, por sua vez, é encapsulado em um datagrama no qual o endereço do servidor é usado como endereço IP de origem, mas cujo endereço de destino é um endereço de *broadcast*; isto permite que outros servidores DHCP recebam a oferta e forneçam uma oferta melhor caso sejam capazes.
3. A estação ingressante recebe uma ou mais propostas e seleciona a melhor delas; então, envia uma mensagem **DHCP**REQUEST para o servidor que forneceu a melhor oferta. Os campos com valores conhecidos são preenchidos. A mensagem é encapsulada em um datagrama de usuário contendo os números de porta iguais aos da primeira mensagem. O datagrama de usuário é encapsulado em um datagrama IP cujo endereço de origem é preenchido com o novo endereço do cliente, mas cujo endereço de destino ainda é mantido como o endereço de *broadcast* para permitir que os outros servidores saibam que suas ofertas não foram aceitas.
4. Finalmente, o servidor selecionado responde com uma mensagem de **DHCP**ACK para o cliente caso o endereço IP oferecido seja válido. Se o servidor não for capaz de manter a sua oferta (por exemplo, se o endereço foi oferecido para outra estação durante esse processo), o servidor envia uma mensagem de **DHCP**NACK e o cliente precisa repetir o processo. Essa mensagem também é transmitida na forma de *broadcast* para que outros servidores saibam que o pedido foi aceito ou rejeitado.

Duas portas bem conhecidas

Dissemos que o DHCP usa duas portas bem conhecidas (68 e 67) em vez de uma porta bem conhecida e outra efêmera. A razão para se escolher a porta bem conhecida (68) e não a porta efêmera para o cliente é que a resposta do servidor para o cliente é enviada por meio de *broadcast*. Lembre-se que um datagrama IP contendo uma mensagem de *broadcast* limitado é entregue a todas as estações na rede. Agora, considere que um cliente DHCP e um cliente de DAYTIME (usado para configuração automática de horário), por exemplo, estão aguardando para receber uma resposta de seu servidor correspondente e ambos acabam acidentalmente utilizando o mesmo número de porta temporária (56017, por exemplo). Ambas as estações recebem a mensagem de resposta do servidor DHCP e entregam a mensagem aos seus clientes. O cliente DHCP processa a mensagem; já o cliente de DAYTIME fica totalmente confuso com uma estranha mensagem recebida. A adoção de um número de porta bem conhecido impede que esse problema venha a ocorrer. A mensagem de resposta do servidor DHCP não é entregue ao cliente de DAYTIME, pois ele está sendo executado na porta de número 56017, e não na 68. Os números de porta temporários e os números de porta bem conhecidos são selecionados de faixas distintas.

O leitor curioso pode se perguntar o que acontece se dois clientes DHCP estiverem sendo executados ao mesmo tempo; isso pode acontecer após uma queda de energia seguida da sua restauração. Nesse caso, as mensagens podem ser distinguidas pelos seus valores de ID da transação, que permitem separar as respostas uma da outra.

Usando FTP

O servidor não envia todas as informações das quais um cliente pode precisar para se juntar à rede. Na mensagem **DHCPACK**, o servidor define o caminho de um arquivo no qual o cliente pode encontrar informações completas, tais como o endereço do servidor DNS. O cliente pode, então, usar um protocolo de transferência de arquivos para obter o restante das informações necessárias.

Controle de erros

O DHCP usa os serviços do UDP, o que não é um protocolo confiável. Para prover controle de erros, o DHCP usa duas estratégias. Primeiramente, o DHCP exige que o UDP use a soma de verificação (*checksum*). Como vimos no Capítulo 3, o uso da soma de verificação no UDP é opcional. Em segundo lugar, o cliente DHCP usa temporizadores e uma política de retransmissão se não receber a resposta correspondente a um pedido DHCP. No entanto, para evitar congestionamento de dados quando várias estações precisam retransmitir um pedido (por exemplo, após uma falha de energia), o DHCP força o cliente a usar um número aleatório para inicializar seus temporizadores.

Estados de transição

Os cenários anteriores nos quais discutimos a operação do DHCP eram muito simples. Para permitir a alocação dinâmica de endereços, o cliente DHCP funciona como uma máquina de estados que realiza transições de um estado para outro, dependendo das mensagens que recebe ou envia. A Figura 4.41 mostra o diagrama de transição de estados do DHCP com seus estados principais.

Quando o cliente DHCP é iniciado pela primeira vez, ele entra no estado INÍCIO (estado de inicialização). O cliente transmite uma mensagem de descoberta via *broadcast*. Quando recebe uma oferta, o cliente vai para o estado SELECIONANDO. Enquanto estiver nesse estado, ele pode receber mais ofertas. Depois de selecionar uma oferta, ele envia uma mensagem de pedido e vai para o estado SOLICITANDO. Se um ACK chegar enquanto o cliente estiver nesse estado, ele vai para o estado VINCULADO e passa a utilizar o endereço IP. Quando o prazo de concessão estiver 50% expirado, o cliente tenta renová-lo movendo-se para o estado RENOVANDO. Se o servidor renovar a concessão, o cliente passa para o estado VINCULADO novamente. Se a concessão não for renovada e o tempo de concessão estiver 87,5% expirado, o cliente passa para o estado REVINCULANDO. Se o servidor aceitar a concessão (uma mensagem de ACK chega ao cliente), o cliente

Figura 4.41 Máquina de estados para o cliente DHCP.

passa para o estado VINCULADO e continua usando o endereço IP; caso contrário, o cliente passa para o estado INÍCIO e solicita outro endereço IP. Observe que o cliente pode usar o endereço IP apenas quando está no estado VINCULADO, RENOVANDO ou REVINCULANDO. O procedimento anterior exige que o cliente use três temporizadores: o *temporizador de renovação* (ajustado para 50% do tempo de concessão), o *temporizador de revinculação* (ajustado para 87,5% do tempo de concessão) e o *temporizador de expiração* (ajustado para o tempo de concessão).

NAT

A distribuição de endereços por ISPs criou um novo problema. Suponha que um ISP tenha concedido um pequeno intervalo de endereços para uma pequena empresa ou residência. Se a empresa crescer ou a família precisar de um intervalo de endereços maior, o ISP pode não ser capaz de atender a essa demanda porque os endereços anteriores e posteriores ao intervalo podem já ter sido atribuídos a outras redes. Na maioria dos casos, entretanto, apenas uma parte dos computadores em uma rede pequena precisa acessar a Internet simultaneamente. Isso significa que o número de endereços atribuídos não precisa coincidir com o número de computadores na rede. Por exemplo, suponha que uma pequena empresa possua 20 computadores, mas o número máximo de computadores que acessam a Internet ao mesmo tempo é apenas 4. A maioria dos computadores está realizando alguma tarefa que não requer acesso à Internet ou eles estão se comunicando uns com os outros. Essa pequena empresa pode usar a pilha de protocolos TCP/IP tanto para sua comunicação interna como para se comunicar com o exterior. A empresa pode usar 20 (ou 25) endereços pertencentes ao bloco de endereços privados (discutidos anteriormente) para a comunicação interna; cinco endereços universais para a comunicação com entidades externas podem ser atribuídos pelo ISP.

Uma tecnologia capaz de fornecer o mapeamento entre os endereços privados e universais e, ao mesmo tempo, dar suporte a redes privadas virtuais, que serão discutidas no Capítulo 10, é a tecnologia de Tradução de Endereços de Rede (NAT – Network Address Translation), que permite que uma organização use um conjunto de endereços particulares para a comunicação interna e um conjunto de endereços globais da Internet (pelo menos um) para a comunicação com o restante do mundo. A organização deve ter uma única conexão à Internet mundial por meio de um roteador com suporte a NAT, que executa o *software* de NAT. A Figura 4.42 mostra uma implementação simples do NAT.

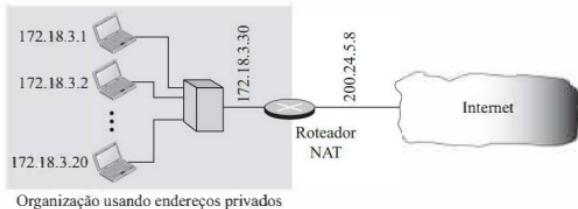



Figura 4.42 NAT

Conforme mostra a figura, a rede privada usa endereços privados. O roteador que conecta a rede interna à Internet usa um endereço privado e um endereço global. A rede privada é invisível para o restante da Internet, que vê apenas o roteador NAT com o endereço 200.24.5.8.

Tradução de endereços

Todos os pacotes de saída passam pelo roteador NAT, que substitui o endereço de origem do pacote pelo endereço de NAT global. Todos os pacotes entrantes também passam pelo roteador NAT, que substitui o endereço de destino do pacote (o endereço global do roteador NAT) pelo endereço privado correspondente. A Figura 4.43 mostra um exemplo de tradução de endereços.

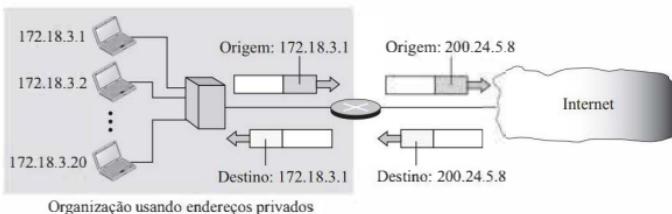



Figura 4.43 Tradução de endereços.

Tabela de tradução

O leitor deve ter notado que a tradução dos endereços de origem no caso de um pacote de saída é bastante simples. Porém, como o roteador NAT sabe o endereço de destino de um pacote vindo da Internet? Pode haver dezenas ou centenas de endereços IP privados, cada um pertencente a uma estação específica. O problema é resolvido se o roteador NAT tiver uma tabela de tradução.

Usando um só endereço IP Em sua forma mais simples, uma tabela de tradução tem apenas duas colunas: o endereço privado e o externo (endereço de destino do pacote). Quando o roteador traduz o endereço de origem do pacote na saída, ele também anota o endereço de destino – o local para onde o pacote está indo. Quando a resposta é recebida do destino, o roteador usa o endereço de origem do pacote (ou seja, o endereço externo) para determinar o endereço privado (ou seja, o endereço interno) para o qual o pacote deve ser entregue. A Figura 4.44 ilustra a ideia.

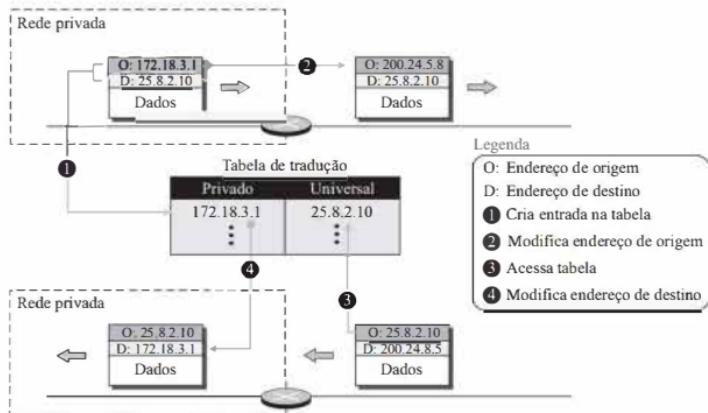



Figura 4.44 Tradução

Nessa estratégia, a comunicação deve ser sempre iniciada pela rede privada. O mecanismo de NAT descrito requer que a rede privada inicie a comunicação. Como veremos, o NAT é usado principalmente por provedores que atribuem um endereço único para um cliente. O cliente, no entanto, pode ser um membro de uma rede privada tendo muitos endereços privados. Nesse caso, a comunicação com a Internet é sempre iniciada pelo cliente, usando um programa-cliente como HTTP, TELNET ou FTP para acessar o programa-servidor correspondente. Por exemplo, quando um e-mail proveniente de fora da rede local é recebido pelo servidor de e-mail do ISP, a mensagem é armazenada na caixa de correio do cliente até que ela seja recuperada por meio de um protocolo como o POP3.

Usando um conjunto de endereços IP O uso de apenas um endereço global pelo roteador NAT permite que apenas uma estação da rede privada acesse um determinado *host* externo. Para eliminar essa restrição, o roteador NAT pode usar um conjunto de endereços globais. Por exemplo, em vez de usar apenas um endereço global (200.24.5.8), o roteador NAT pode usar quatro endereços (200.24.5.8, 200.24.5.9, 200.24.5.10 e 200.24.5.11). Nesse caso, quatro estações da rede privada podem se comunicar com o mesmo *host* externo simultaneamente, pois cada par de endereços define uma conexão distinta. No entanto, existem ainda algumas desvantagens. Não é possível criar mais do que quatro conexões para o mesmo destino. Nenhuma estação da rede privada pode acessar dois programas servidores externos (por exemplo, HTTP e TELNET) ao mesmo tempo. E, do mesmo modo, duas estações da rede privada não podem acessar o mesmo programa-servidor externo (por exemplo, HTTP ou TELNET) simultaneamente.

Usando tanto endereços IP como endereços de porta Para permitir uma relação muitos para muitos entre estações da rede privada e programas-servidores externos, precisamos de mais informações na tabela de tradução. Por exemplo, considere que duas estações em uma rede privada, de endereços 172.18.3.1 e 172.18.3.2, precisam acessar o servidor HTTP no *host* externo 25.8.3.2. Se a tabela de tradução tiver cinco colunas em vez de duas, incluindo os endereços das portas de origem e de destino e também o protocolo da camada de transporte, a ambiguidade é eliminada. A Tabela 4.1 mostra um exemplo de uma tabela desse tipo.

Tabela 4.1 Tabela de tradução com cinco colunas.

Endereço privado	Porta privada	Endereço externo	Porta externa	Protocolo de transporte
172.18.3.1	1400	25.8.3.2	80	TCP
172.18.3.2	1401	25.8.3.2	80	TCP
⋮	⋮	⋮	⋮	⋮

Note que, quando a resposta do HTTP retorna, a combinação do endereço de origem (25.8.3.2) e do endereço de porta do destino (1401) determina a estação da rede privada para o qual a resposta deve ser entregue. Perceba também que, para que essa tradução funcione, os endereços das portas efêmeras (1400 e 1401) devem ser únicos.

4.2.3 Encaminhamento de pacotes IP

Discutimos o conceito de transmissão na camada de rede no início deste capítulo. Nesta seção, vamos estender esse conceito para incluir o papel dos endereços IP no encaminhamento. Conforme discutimos anteriormente, o conceito de encaminhamento significa colocar o pacote na rota para seu destino. Como a Internet atual é composta por uma combinação de redes e conexões, o encaminhamento equivale a entregar o pacote para o próximo salto (que pode ser o destino final do pacote ou um dispositivo de conexão intermediário). Embora o protocolo IP tenha sido originalmente concebido como um protocolo não orientado à conexão, a tendência atual é torná-lo um protocolo orientado à conexão. Discutiremos ambos os casos.

Quando o IP é usado como um protocolo não orientado à conexão, o encaminhamento é baseado no endereço de destino do datagrama IP; já quando o IP é usado como um protocolo orientado à conexão, o encaminhamento é baseado no rótulo anexado a um datagrama IP.

Encaminhamento baseado no endereço de destino

Primeiramente, discutiremos o encaminhamento baseado no endereço de destino. Essa é uma abordagem tradicional, a mais comum atualmente, na qual o encaminhamento requer que uma estação ou roteador tenha uma tabela de roteamento. Quando uma estação tem um pacote para enviar ou quando um roteador recebe um pacote a ser encaminhado, eles verificam essa tabela em busca do próximo salto para o qual o pacote deve ser entregue.

No endereçamento sem classes, o espaço de endereços inteiro constitui uma entidade; não há classes. Isto significa que o encaminhamento requer uma linha de informação para cada bloco envolvido. A tabela precisa ser verificada com base no endereço de rede (primeiro endereço do bloco). Infelizmente, o endereço de destino no pacote não fornece qualquer pista sobre o endereço de rede. Para resolver esse problema, precisamos incluir a máscara (/n) na tabela. Em outras palavras, uma tabela de roteamento sem classes deve incluir quatro informações: a máscara, o endereço de rede, o número da interface e também o endereço IP do próximo roteador (necessário para localizar o endereço da camada de enlace do próximo salto, conforme discutiremos no Capítulo 5). No entanto, muitas vezes vemos na literatura que as duas primeiras informações são combinadas. Por exemplo, se $n = 26$ e o endereço de rede é 180.70.65.192, então pode-se combinar as duas informações na forma 180.70.65.192/**26**. A Figura 4.45 mostra um módulo de encaminhamento e uma tabela de roteamento simples para um roteador tendo apenas três interfaces.

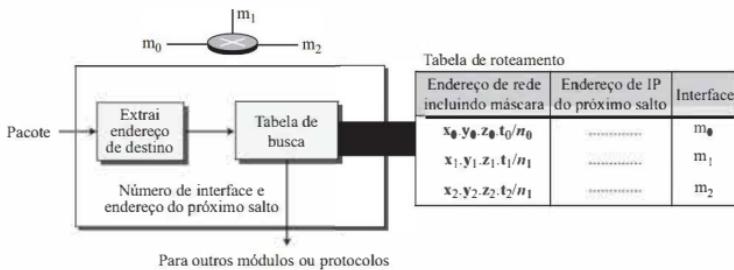



Figura 4.45 Módulo simplificado de encaminhamento no endereçamento sem classes.

A tarefa do módulo de encaminhamento é fazer buscas na tabela, linha por linha. Em cada linha, os *n bits* mais à esquerda do endereço de destino (prefixo) são mantidos e o restante dos *bits* (sufixo) são fixados em 0. Se o endereço resultante (denominado *endereço de rede*) coincidir com o endereço na primeira coluna, as informações nas próximas duas colunas são extraídas; caso contrário, a busca continua. Normalmente, a última linha tem um valor-padrão na sua primeira coluna (não mostrada na figura), indicando todos os endereços de destino que não correspondem às linhas anteriores.

Algumas vezes, a literatura mostra explicitamente o valor dos *n bits* mais à esquerda que devem ser comparados com os *n bits* mais à esquerda do endereço de destino. O conceito é idêntico, mas a representação é diferente. Por exemplo, em vez de fornecer a combinação endereço-máscara de 180.70.65.192/**26**, podemos fornecer o valor dos 26 *bits* mais à esquerda conforme mostrado a seguir.

10110100 01000110 01000001 11

Perceba que ainda precisamos usar um algoritmo para determinar o prefixo e compará-lo com o padrão de *bits*. Em outras palavras, o algoritmo ainda é necessário, mas a representação é diferente. Usamos esse formato em nossas tabelas de roteamento apresentadas nos exercícios no final do capítulo, quando empregamos espaços de endereços menores apenas para praticar esses conceitos.

Exemplo 4.7

Crie uma tabela de roteamento para o roteador R1 utilizando a configuração dada na Figura 4.46.

Solução

A Tabela 4.2 mostra a tabela correspondente.

Tabela 4.2 Tabela de roteamento para o roteador R1 da Figura 4.46.

Endereço de rede/máscara	Próximo salto	Interface
180.70.65.192/**26**	—	m2
180.70.65.128/**25**	—	m0
201.4.22.0/**24**	—	m3
201.4.16.0/**22**	—	m1
Padrão	180.70.65.200	m2

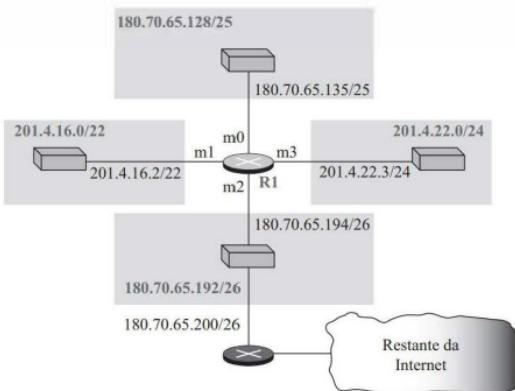



Figura 4.46 Configuração para o Exemplo 4.7.

Exemplo 4.8

Em vez da Tabela 4.2, podemos usar a Tabela 4.3, na qual o endereço de rede/máscara é dado em *bits*.

Tabela 4.3 Tabela de roteamento para o roteador R1 da Figura 4.46 usando prefixos em *bits*.

Bits mais à esquerda no endereço de destino	Próximo salto	Interface
10110100 01000110 01000001 11	—	m2
10110100 01000110 01000001 1	—	m0
11001001 00000100 00011100	—	m3
11001001 00000100 000100	—	m1
Padrão	180.70.65.200	m2

Quando um pacote chega e seus 26 *bits* mais à esquerda no endereço de destino coincidem com os *bits* da primeira linha, o pacote sai pela interface m2. Quando um pacote chega e seus 25 *bits* mais à esquerda no endereço de destino coincidem com os *bits* da segunda linha, o pacote sai pela interface m0 e assim por diante. A tabela mostra claramente que a primeira linha tem o prefixo mais longo e que a quarta linha tem o prefixo mais curto. Um prefixo mais longo indica uma faixa menor de endereços; um mais curto, uma faixa maior de endereços.

Exemplo 4.9

Mostre o processo de encaminhamento quando um pacote cujo endereço de destino 180.70.65.140 chega a R1 na Figura 4.46.

Solução

O roteador executa os seguintes passos:

1. A primeira máscara (/26) é aplicada ao endereço de destino. O resultado é 180.70.65.128, que não coincide com o endereço de rede correspondente.
2. A segunda máscara (/25) é aplicada ao endereço de destino. O resultado é 180.70.65.128, que coincide com o endereço de rede correspondente. O endereço do próximo salto e o número da interface m0 são extraídos para encaminhar o pacote (ver Capítulo 5).

Agregação de endereços

Quando usamos endereçamento com classes, existe apenas uma entrada na tabela de roteamento para cada local externo à organização. A entrada define o local mesmo quando este é composto por sub-redes. Quando um pacote chega ao roteador, este verifica a entrada correspondente e encaminha o pacote de acordo com a sua tabela. Quando usamos endereçamento sem classes, é provável que o número de entradas da tabela de roteamento aumente, porque o objetivo do endereçamento sem classes é dividir o espaço de endereços inteiro em blocos gerenciáveis. O aumento no tamanho da tabela resulta em um aumento no tempo necessário para realizar buscas na tabela. Para aliviar esse problema, a ideia de agregação de endereços foi concebida. Na Figura 4.47, temos dois roteadores.

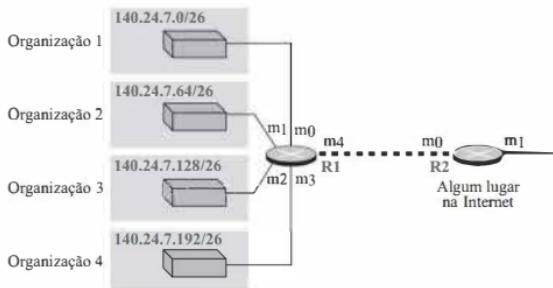



Tabela de roteamento para R1

Endereço de rede/máscara	Endereço do próximo salto	Interface
140.24.7.0/26	----------	m0
140.24.7.64/26	----------	m1
140.24.7.128/26	----------	m2
140.24.7.192/26	----------	m3
0.0.0.0/0	endereço de R2	m4

Tabela de roteamento para R2

Endereço de rede/máscara	Endereço do próximo salto	Interface
140.24.7.0/24	----------	m0
0.0.0.0/0	roteador-padrão	m1

Figura 4.47 Agregação de endereços.

R1 está conectado a redes de quatro organizações que usam, cada uma, 64 endereços. R2 está em algum lugar distante de R1, que tem uma tabela de roteamento maior porque cada pacote deve ser corretamente roteado para a organização apropriada. R2, por outro lado, pode ter uma tabela de roteamento bem pequena. Para R2, qualquer pacote cujo destino esteja entre 140.24.7.0 e 140.24.7.255 é entregue pela interface m0, independentemente do número da organização. Isto é chamado agregação de endereços porque os blocos de endereços para as quatro organizações são agregados em um bloco maior. R2 teria uma tabela de roteamento maior se cada organização tivesse endereços que não pudessem ser agregados em um bloco.

Correspondência da máscara mais longa

O que acontece se uma das organizações na figura anterior não estiver geograficamente próxima às outras três? Por exemplo, se a organização 4 não puder ser conectada ao roteador R1 por alguma razão, ainda podemos usar a ideia de agregação de endereços e ainda atribuir o bloco 140.24.7.192/26 à organização 4? A resposta é sim, porque o roteamento no endereçamento sem classes usa um outro princípio, o da ***correspondência da máscara mais longa***. Esse princípio dita que a tabela de roteamento é ordenada da máscara mais longa para a mais curta. Em outras palavras, se houver três máscaras, /27, /26 e /24, a máscara /27 deve ser a primeira entrada na tabela e a /24 deve ser a última. Vejamos se esse princípio resolve a situação em que a organização 4 encontra-se separada das outras três organizações. A Figura 4.48 ilustra essa situação.

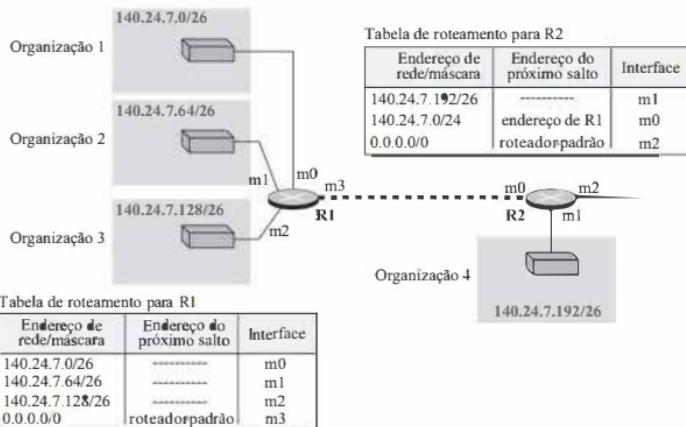



Figura 4.48 Correspondência da máscara mais longa.

Considere que um pacote chegue ao roteador R2 e que ele seja endereçado à organização 4, tendo como endereço de destino 140.24.7.200. A primeira máscara no roteador R2 é aplicada, resultando no endereço de rede 140.24.7.192. O pacote é roteado corretamente por meio da interface m1 e chega à organização 4. Se, no entanto, a tabela de roteamento não fosse armazenada com o prefixo mais longo na sua primeira linha, a aplicação da máscara /24 resultaria no encaminhamento incorreto do pacote para o roteador R1.

Roteamento hierárquico

Para resolver o problema de tabelas de roteamento gigantescas, podemos criar uma forma de hierarquia nessas tabelas. No Capítulo 1, mencionamos que a Internet atual apresenta uma forma de hierarquia. Dissemos que a Internet é dividida em ISPs de *backbone* e ISPs nacionais. Os ISPs nacionais são divididos em ISPs regionais, enquanto os ISPs regionais são divididos em ISPs locais. Se a tabela de roteamento tiver uma forma de hierarquia tal como a arquitetura da Internet, a tabela de roteamento pode diminuir de tamanho.

Tomemos o caso de um ISP local. Um ISP local pode receber um único, porém grande, bloco de endereços com um determinado comprimento de prefixo. O ISP local pode dividir esse bloco em blocos menores de tamanhos diferentes e atribuí-los a usuários individuais e organizações, tanto

grandes como pequenas. Se o bloco atribuído ao ISP local começa com a.b.c.d/n, o ISP pode criar blocos começando com e.f.g.h/m, onde m pode variar dependendo do cliente, mas é sempre maior do que n.

Como isso reduz o tamanho da tabela de roteamento? O restante da Internet não precisa saber sobre essa divisão. Todos os clientes do ISP local são definidos como a.b.c.d/n para o restante da Internet. Cada pacote destinado a um dos endereços nesse grande bloco é encaminhado para o ISP local. Existe apenas uma entrada em cada roteador no mundo para todos esses clientes, que pertencem ao mesmo grupo. Obviamente, internamente ao ISP local, o roteador deve reconhecer os sub-blocos e encaminhar o pacote para o cliente correto. Se um dos clientes for uma grande organização, ela própria pode criar outro nível de hierarquia usando sub-redes e dividindo seu sub-bloco em sub-blocos menores (ou subsub-blocos). No roteamento sem classes, os níveis de hierarquia são ilimitados contanto que respeitemos as regras do endereçamento sem classes.

Exemplo 4.10

Como exemplo de roteamento hierárquico, vamos considerar a Figura 4.49. Um ISP regional recebeu 16.384 endereços começando em 120.14.64.0. O ISP regional decidiu dividir esse bloco em quatro sub-blocos, cada qual com 4.096 endereços. Três desses sub-blocos são atribuídos a três ISPs locais, enquanto o segundo sub-bloco é reservado para utilização futura. Perceba que a máscara para cada bloco é /20 porque o bloco original com máscara /18 é dividido em quatro sub-blocos.

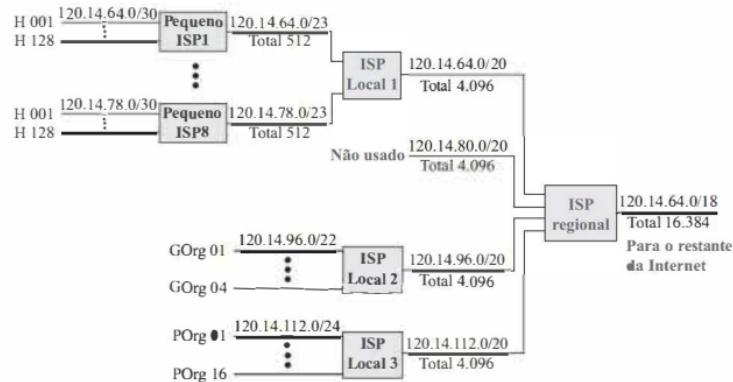



Figura 4.49 Roteamento hierárquico com ISPs.

O primeiro ISP local dividiu o bloco a ele alocado em oito sub-blocos menores e atribuiu cada um deles a um pequeno ISP. Cada pequeno ISP fornece serviços para 128 residências (R001 a R128), cada uma com quatro endereços. Perceba que a máscara para cada pequeno ISP é agora /23 porque o bloco recebido por ele foi dividido em oito blocos. Cada residência recebe uma máscara /30, porque uma casa tem apenas quatro endereços ($2^{32-30} = 4$). O segundo ISP local dividiu seu bloco em quatro sub-blocos e atribuiu os endereços a quatro organizações de grande porte (GOrg01 a GOrg04). Perceba que cada uma dessas organizações tem 1.024 endereços e sua máscara é /22.

O terceiro ISP local dividiu seu bloco em 16 sub-blocos e atribuiu cada um deles a uma organização de pequeno porte (POrg01 a POrg16). Cada organização de pequeno porte tem 256 endereços

e sua máscara é /24. Existe uma forma de hierarquia nessa configuração. Todos os roteadores nessa internet enviam pacotes com endereço de destino entre 120.14.127.255 e 120.14.64.0 para o ISP regional, que envia todos os pacotes tendo endereço de destino entre 120.14.64.0 e 120.14.79.255 para os ISP Local 1. O ISP1 local envia todos os pacotes cujo endereço de destino esteja entre 120.14.64.0 a 120.14.64.3 para R001.

Roteamento geográfico

Para reduzir o tamanho da tabela de roteamento ainda mais, precisamos estender o roteamento hierárquico de modo a incluir a noção de roteamento geográfico. Precisamos dividir o espaço de endereços completo em alguns poucos blocos de grandes dimensões. Alocamos um bloco para a América, um para a Europa, um para a Ásia, outro para a África, e assim por diante. Os roteadores dos ISPs fora da Europa terão uma única linha para os pacotes destinados à Europa em suas tabelas de roteamento. Os roteadores de ISPs fora da América terão uma única linha para os pacotes destinados à América em suas tabelas de roteamento e assim por diante.

Algoritmos de busca na tabela de roteamento

No endereçamento sem classes, não existem informações sobre a rede no endereço de destino. O método de busca mais simples, porém não o mais eficiente, é o de correspondência da máscara mais longa (conforme discutimos anteriormente). A tabela de roteamento pode ser dividida em grupos, um para cada prefixo. O roteador tenta primeiro o prefixo mais longo. Se o endereço de destino for encontrado nesse grupo, a pesquisa está concluída. Se o endereço não for encontrado, o próximo prefixo é utilizado e assim por diante. É óbvio que esse tipo de pesquisa demora muito tempo.

Uma solução é alterar a estrutura de dados usada para a busca. Em vez de usar uma lista, outras estruturas de dados (como uma árvore ou uma árvore binária) podem ser empregadas. Um candidato é um tipo especial de árvore conhecida como árvore *trie**. Entretanto, tal discussão extrapola o escopo deste livro.

Roteamento baseado em rótulos

Na década de 1980, um esforço foi iniciado para, de alguma forma, fazer com que o IP passasse a se comportar como um protocolo orientado à conexão no qual o roteamento seria substituído pela comutação. Conforme já discutimos neste capítulo, em uma rede não orientada à conexão (abordagem de datagramas), os roteadores encaminham pacotes com base no endereço de destino presente no cabeçalho de cada pacote. Por outro lado, em uma rede orientada à conexão (abordagem de circuitos virtuais), os comutadores encaminham pacotes com base no rótulo vinculado a cada pacote. O roteamento é normalmente baseado na busca pelos conteúdos de uma tabela; já a comutação pode ser feita acessando um índice de uma tabela. Em outras palavras, a tarefa de roteamento envolve buscas; já a comutação envolve acessos a índices.

Exemplo 4.11

A Figura 4.50 mostra um exemplo simples de busca na tabela de roteamento usando o algoritmo de correspondência da máscara mais longa. Apesar de existirem alguns algoritmos mais eficientes atualmente, o princípio básico permanece o mesmo.

Quando o algoritmo de encaminhamento obtém o endereço de destino do pacote, ele precisa verificar a coluna contendo as máscaras de rede. Para cada linha, ele precisa aplicar a máscara para encontrar o endereço de rede do destino. Em seguida, deve verificar os endereços de rede na tabela até encontrar uma correspondência. O roteador extrai, então, o endereço do próximo salto e o número da interface a ser entregue para a camada de enlace de dados.

* N. de T.: Também conhecida como árvore de prefixos.

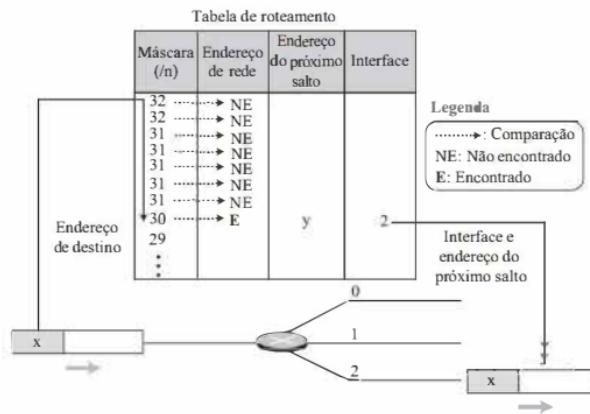
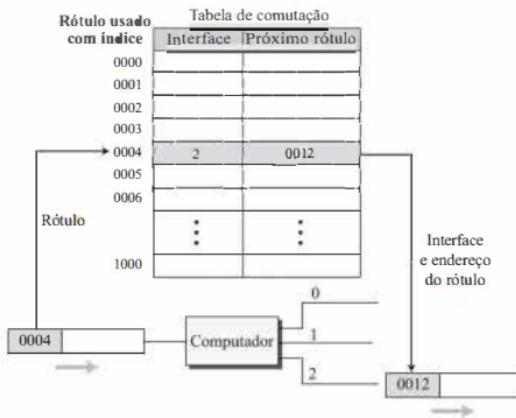



Figura 4.50 Esquema para o Exemplo 4.11: roteamento baseado no endereço de destino.

Exemplo 4.12

A Figura 4.51 mostra um exemplo simples de uso de um rótulo para acessar uma tabela de comutação. Dado que os rótulos são utilizados como índices para a tabela, pode-se encontrar a informação na tabela de forma imediata.

Figura 4.51 Esquema para o Exemplo 4.12: roteamento baseado em rótulos.

Comutação de Rótulos Multiprotocolo

Durante a década de 1980, diversos fabricantes criaram roteadores que implementavam a tecnologia de comutação. Mais tarde, a IETF aprovou um padrão denominado Comutação de Rótulos Multiprotocolo (MPLS – Multi-Protocol Label Switching). De acordo com ele, alguns roteadores convencionais da Internet podem ser substituídos por roteadores MPLS, os quais podem se comportar como um roteador e como um *switch*. Quando eles se comportam como roteadores, o MPLS pode encaminhar o pacote com base no endereço de destino; quando se comportam como *switches*, o protocolo pode encaminhar pacotes com base em seu rótulo.

Um novo cabeçalho

Para simular a comutação orientada à conexão usando um protocolo como o IP, a primeira coisa necessária é adicionar um campo ao pacote que carrega o rótulo discutido mais adiante. O formato do pacote IPv4 não permite essa extensão (embora esse campo esteja presente no formato dos pacotes IPv6, conforme veremos mais tarde). A solução é encapsular o pacote IPv4 em um pacote MPLS (como se o MPLS fosse uma camada entre a camada de enlace de dados e a de rede). O pacote IP inteiro é encapsulado como a carga útil (*payload*) em um pacote MPLS, ao qual é adicionado um cabeçalho MPLS. A Figura 4.52 mostra o encapsulamento.

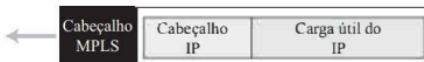



Figura 4.52 Cabeçalho MPLS adicionado a um pacote IP.

O cabeçalho MPLS é, na verdade, uma pilha de subcabeçalhos usada para comutação hierárquica multinível, conforme veremos em breve. A Figura 4.53 mostra o formato de um cabeçalho MPLS em que cada subcabeçalho tem 32 *bits* (4 *bytes*) de comprimento.

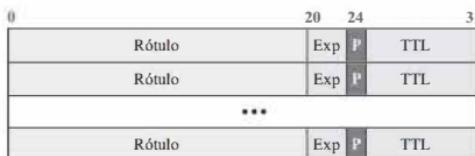



Figura 4.53 Cabeçalho MPLS, composto por uma pilha de rótulos.

A seguir, fornecemos uma breve descrição de cada campo:

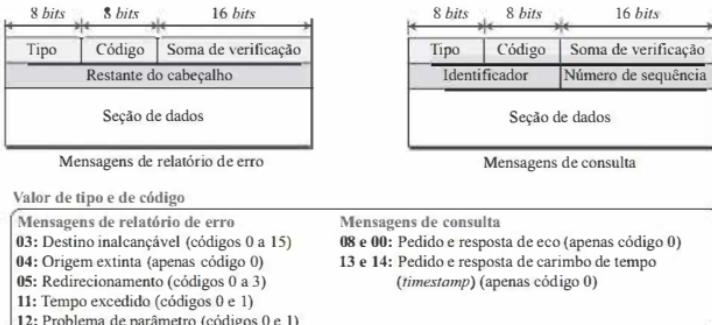
- **Rótulo.** Esse campo de 20 *bits* define o rótulo que é usado para indexar a tabela de roteamento no roteador.
- **Exp.** Esse campo de 3 *bits* é reservado para fins experimentais.
- **P.** O campo de pilha tem um só *bit* e define a situação do subcabeçalho na pilha de rótulos. Quando o *bit* é 1, isto significa que o cabeçalho é o último da pilha.
- **TTL.** Esse campo de 8 *bits* é semelhante ao campo TTL no datagrama IP. Cada roteador visitado decrementa o valor desse campo. Quando ele chega a zero, o pacote é descartado para evitar que ele trafegue indefinidamente.

Comutação hierárquica

Uma pilha de rótulos no MPLS é o que permite a comutação hierárquica. Isto é semelhante ao roteamento hierárquico convencional. Por exemplo, se um pacote tem dois rótulos, o rótulo mais alto da pilha pode ser usado para encaminhar o pacote por comutadores externos a uma organização; já o rótulo abaixo dele pode ser usado para rotear o pacote dentro da organização até que ele chegue à sub-rede de destino.

4.2.4 ICMPv4

O IPv4 não dispõe de qualquer mecanismo para reportar ou corrigir erros. O que acontece se algo der errado? O que acontece se um roteador for obrigado a descartar um datagrama porque ele não pode encontrar uma rota para o destino final desse datagrama, ou porque o campo TTL atingiu o valor zero? O que acontece se a estação de destino do pacote tiver que descartar os fragmentos de um datagrama recebido porque ela não recebeu todos os fragmentos dentro de um limite de tempo pré-estabelecido? Esses são exemplos de situações em que um erro ocorreu, mas o protocolo IP não dispõe de qualquer mecanismo intrínseco para notificar a estação de origem desse fato.

O protocolo IP também carece de um mecanismo para consultas de gerenciamento e referentes a estações na rede. Uma estação às vezes precisa determinar se um roteador ou outra estação estão ativos. E às vezes um gerente de rede precisa de informações sobre o desempenho de uma estação ou roteador.

O **Protocolo de Mensagens de Controle da Internet versão 4** (ICMPv4 – Internet Control Message Protocol version 4) foi concebido para compensar as duas deficiências anteriores. Ele é um parceiro do protocolo IP. O ICMP em si é um protocolo da camada de rede. No entanto, suas mensagens não são passadas diretamente para a camada de enlace de dados, como seria esperado. Na verdade, as mensagens são primeiramente encapsuladas em datagramas IP antes de irem para a camada inferior. Quando um datagrama IP encapsula uma mensagem ICMP, o valor do campo de protocolo no datagrama IP é fixado em 1 para indicar que a carga útil desse datagrama é uma mensagem ICMP.

Mensagens

As mensagens ICMPv4 são divididas em duas grandes categorias: *mensagens de relatório de erro* e *mensagens de consulta*. As mensagens de relatório de erro relatam os problemas que um roteador ou uma estação (de destino) podem encontrar quando processam um pacote IP. As mensagens de consulta, que aparecem em pares, ajudam uma estação ou um gerente de rede a obter informações específicas provenientes de um roteador ou outra estação. Por exemplo, as estações podem determinar quem são as estações ou os roteadores na vizinhança de sua rede e os roteadores podem ajudar um nó a redirecionar suas mensagens.

Formato das mensagens

Uma mensagem ICMPv4 tem um cabeçalho de 8 *bytes* e uma seção de dados de tamanho variável. Embora o formato geral do cabeçalho seja diferente para cada tipo de mensagem, os primeiros 4 *bytes* são comuns a todas elas. Conforme mostrado na Figura 4.54, o primeiro campo, o tipo de ICMP, define o tipo da mensagem. O campo de código especifica a razão particular para o tipo de mensagem. O último campo em comum é o campo de soma de verificação (*checksum*). O restante do cabeçalho é específico para cada tipo de mensagem. A seção de dados nas mensagens de relatório de erro transporta as informações necessárias para determinar o pacote original com o qual ocorreu o erro. Em mensagens de consulta, a seção de dados contém informações extras que dependem do tipo de consulta.

Valor de tipo e de código

Mensagens de relatório de erro	**Mensagens de consulta**
03: Destino inalcançável (códigos 0 a 15)	**08 e 00:** Pedido e resposta de eco (apenas código 0)
04: Origem extinta (apenas código 0)	**13 e 14:** Pedido e resposta de carimbo de tempo (*timestamp*) (apenas código 0)
05: Redirecionamento (códigos 0 a 3)	
11: Tempo excedido (códigos 0 e 1)	
12: Problema de parâmetro (códigos 0 e 1)	

Nota: Consulte o *site* do livro para obter mais explicações sobre os valores dos códigos.

Figura 4.54 Formato genérico das mensagens ICMP.

Mensagens de relatório de erro

Como o IP é um protocolo não confiável, uma das principais responsabilidades do ICMP é relatar alguns erros que podem ocorrer durante o processamento de datagramas IP. O ICMP não corrige erros: ele simplesmente os relata. A correção de erros é deixada para os protocolos de camadas superiores. Mensagens de erro são sempre enviadas para a fonte original do pacote, pois a única informação disponível no datagrama sobre sua rota são os endereços IP de origem e de destino. O ICMP usa o endereço IP de origem para enviar a mensagem de erro para a fonte (originador) do datagrama. Para simplificar o processo de relatar erros, o ICMP segue algumas regras para as mensagens de relatório. Em primeiro lugar, nenhuma mensagem de relatório de erro é gerada para datagramas contendo um endereço *multicast* ou endereço especial (como *esta estação* ou *loopback*). Em segundo lugar, nenhuma mensagem de relatório de erro ICMP é gerada em resposta a um datagrama que transporta outra mensagem ICMP deste tipo. Em terceiro lugar, nenhuma mensagem de relatório de erro ICMP é gerada para um datagrama fragmentado quando ele não for o primeiro fragmento.

Todas as mensagens de relatório de erro contêm uma seção de dados que inclui o cabeçalho IP do datagrama original acompanhado dos 8 primeiros *bytes* de dados daquele datagrama. O cabeçalho do datagrama original é adicionado para fornecer à fonte original, que recebe a mensagem de erro, informações sobre o datagrama em si. Os 8 *bytes* de dados estão incluídos porque os primeiros 8 *bytes* fornecem informações sobre os números das portas (UDP e TCP) e sobre número de sequência (TCP). Essas informações são necessárias para que a origem do pacote possa informar os protocolos da camada superior (TCP ou UDP) sobre o erro.

A mensagem de relatório de erro mais amplamente utilizada é a de *destino inalcançável* (tipo 3), que usa códigos diferentes (0 a 15) para definir o tipo de erro e a razão pela qual um datagrama não alcançou seu destino final. Por exemplo, o código 0 informa à origem que um *host* está inacessível. Isto pode acontecer, por exemplo, quando usamos o protocolo HTTP para acessar uma página Web, mas o servidor não está respondendo. A mensagem "*host* de destino inalcançável" é criada e enviada de volta para a fonte.

Outra mensagem de relatório de erro é a chamada *origem extinta*, ou *source quench* (tipo 4), que informa o emissor dos dados que a rede apresenta congestionamento e que o datagrama foi descartado; o emissor precisa reduzir a taxa de envio dos próximos datagramas. Em outras palavras, ao usar esse tipo de mensagem, o ICMP adiciona um tipo de mecanismo de controle de congestionamento ao protocolo IP.

A *mensagem de redirecionamento* (tipo 5) é utilizada quando a fonte usa um roteador errado para enviar sua mensagem. Ele redireciona a mensagem para o roteador apropriado, mas informa à origem que ela deve alterar o seu roteador-padrão para futuras mensagens. O endereço IP do roteador-padrão é enviado na mensagem.

Discutimos o propósito do campo Tempo de Vida (TTL – Time-To-Live) no datagrama IP e explicamos que ele impede que um datagrama circule indefinidamente na Internet. Quando o valor do TTL chega a 0, o datagrama é descartado pelo roteador sendo visitado e uma mensagem de *tempo excedido* (tipo 11) com o código 0 é enviada para a fonte de modo a informá-la sobre o ocorrido. A mensagem de tempo excedido (com o código 1) também pode ser enviada quando nem todos os fragmentos de um datagrama chegam ao destino dentro de um tempo pré-estabelecido.

Finalmente, uma mensagem de *problema de parâmetro* (tipo 12) pode ser enviada quando há um problema no cabeçalho de um datagrama (código 0) ou quando algumas opções estão ausentes ou não podem ser interpretadas (código 1).

Mensagens de consulta

Curiosamente, as mensagens de consulta no ICMP podem ser usadas de forma independente, sem estarem relacionadas a um datagrama IP. Porém, obviamente, uma mensagem de consulta precisa ser encapsulada em um datagrama, o qual carrega os dados da mensagem. Mensagens de consulta são usadas para verificar ou testar a disponibilidade de estações ou roteadores na internet, para encontrar o tempo de ida e volta (RTT) ou só de ida de um datagrama IP entre dois dispositivos e até mesmo para saber se os relógios em dois dispositivos estão sincronizados. Naturalmente, as mensagens de consulta aparecem em pares: pedido e resposta.

O par de mensagens de *pedido de eco* (tipo 8) e *resposta de eco* (tipo 0) é usado por uma estação ou um roteador para testar a disponibilidade de uma outra estação ou de um roteador. A estação (ou o roteador) envia uma mensagem de pedido de eco a outra estação ou roteador; caso a máquina-alvo esteja ativa, ela responde com uma mensagem de resposta de eco. Veremos brevemente as aplicações desse par de mensagens em duas ferramentas de depuração: *ping* e *traceroute*.

O par de mensagens de *pedido de carimbo de tempo* (tipo 13) e *resposta de carimbo de tempo* (tipo 14) é usado para encontrar o Tempo de Ida e Volta (RTT – Round-Trip-Time) entre dois dispositivos, ou para verificar se os relógios em dois dispositivos estão sincronizados. A mensagem de pedido de carimbo de tempo (*timestamp*) carrega um número de 32 *bits* que define o instante em que a mensagem foi enviada. A resposta de carimbo de tempo reenvia esse número, mas também inclui dois novos números de 32 *bits* representando o instante em que o pedido foi recebido e o instante em que a resposta foi enviada. Se todos os carimbos de tempo representam o tempo universal, o remetente pode calcular o tempo de ida e volta e o tempo só de ida entre ele e o destinatário da mensagem.

Exemplo 4.13

Uma das ferramentas que uma estação pode usar para testar se outra estação está ativa é o programa *ping*. O programa *ping* usa as mensagens ICMP de pedido e resposta de eco. Uma estação pode enviar uma mensagem de pedido de eco (tipo 8, código 0) para outra estação que, caso esteja disponível, pode enviar de volta uma mensagem de resposta de eco (tipo 0, código 0). Até certo ponto, o programa *ping* também pode mensurar a confiabilidade e o congestionamento do roteador entre as duas estações, enviando um conjunto de mensagens de pedido e resposta.

A seguir, mostramos como pode-se enviar uma mensagem de *ping* para o *site* umauniversidade.edu. Preenchemos o campo de identificação na mensagem de pedido de eco e iniciamos o número de sequência em 0; esse número é incrementado em um cada vez que uma nova mensagem é enviada. Observe que o programa *ping* pode calcular o RTT. Ele insere o valor do instante de envio na seção de dados da mensagem. Quando o pacote de resposta retorna, o programa subtrai o seu instante de chegada do instante de saída do pedido para determinar o RTT.

```
$ ping umauniversidade.edu
PING umauniversidade.edu (152.181.8.3) 56 (84) bytes de dados.
64 bytes de umauniversidade.edu (152.181.8.3): icmp_seq=0    ttl=62    tempo=1.91 ms
64 bytes de umauniversidade.edu (152.181.8.3): icmp_seq=1    ttl=62    tempo=2.04 ms
64 bytes de umauniversidade.edu (152.181.8.3): icmp_seq=2    ttl=62    tempo=1.90 ms
64 bytes de umauniversidade.edu (152.181.8.3): icmp_seq=3    ttl=62    tempo=1.97 ms
64 bytes de umauniversidade.edu (152.181.8.3): icmp_seq=4    ttl=62    tempo=1.93 ms
64 bytes de umauniversidade.edu (152.181.8.3): icmp_seq=5    ttl=62    tempo=2.00 ms
--- estatísticas para umauniversidade.edu ---
6 pacotes transmitidos, 6 recebidos, 0% de perda de pacotes
rtt min/med/max = 1.90/1.95/2.04 ms
```

Exemplo 4.14

O programa *traceroute*[*] no UNIX ou *tracert* no Windows pode ser usado para rastrear o caminho de um pacote da origem até o destino. É possível determinar os endereços IP de todos os roteadores que são visitados ao longo desse caminho. O programa é normalmente configurado para verificar um máximo de 30 saltos (roteadores) a serem visitados. O número de saltos na Internet costuma ser menor que esse valor. O programa *traceroute* é diferente do programa *ping*, que faz uso de duas mensagens de consulta; já o programa *traceroute* faz uso de duas mensagens de relatório de erro: de tempo excedido e de destino inalcançável. O *traceroute* é um programa da camada de aplicação, mas somente o programa-cliente é necessário porque, como veremos, as mensagens enviadas pelo programa-cliente nunca chegam à camada de aplicação da estação de destino. Em outras palavras, não existe um programa servidor de *traceroute*. O programa *traceroute* é encapsulado em um datagrama de usuário UDP, mas usa intencionalmente um número de porta não disponível no destino. Se houver n roteadores no caminho até o destino, o *traceroute* envia $(n + 1)$ mensagens. As primeiras n mensagens são descartadas pelos n roteadores, uma por cada roteador; a última mensagem é descartada pela estação de destino. O programa-cliente do *traceroute* usa as $(n + 1)$ mensagens de relatório de erro ICMP recebidas para determinar o caminho entre as estações. Mostraremos brevemente que o *traceroute* não precisa saber o valor de n, que é determinado automaticamente. A Figura 4.55 mostra um exemplo no qual $n = 3$.

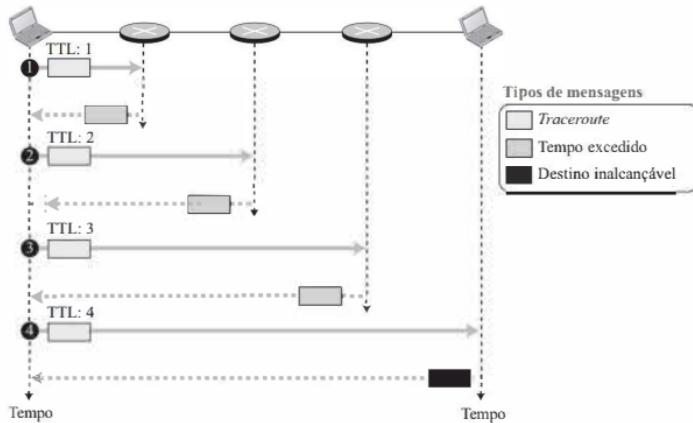



Figura 4.55 Exemplo do programa *traceroute*.

[*] N. de T.: Em algumas distribuições Linux, o programa *traceroute* foi substituído pelo programa *tracepath*.

A primeira mensagem do *traceroute* é enviada com o valor do TTL fixado em 1; ela é descartada no primeiro roteador, que envia para a origem uma mensagem de relatório de erro ICMP do tipo tempo excedido, a partir da qual o *traceroute* pode determinar o endereço IP do primeiro roteador (o endereço IP de origem na mensagem de relatório de erro) e o nome desse roteador (na seção de dados da mensagem). A segunda mensagem do *traceroute* é enviada com TTL fixado em 2, o que permite determinar o endereço IP e o nome do segundo roteador. Da mesma forma, a terceira mensagem permite determinar as informações sobre o roteador 3. A quarta mensagem, entretanto, atinge a estação de destino. Ela também descarta a mensagem, mas por outro motivo: a estação de destino não é capaz de encontrar o número de porta especificado no datagrama de usuário UDP. Dessa vez, é enviada uma mensagem ICMP diferente: a mensagem de destino inalcançável, com código 3, para indicar que o número da porta não foi encontrado. Ao receber a mensagem ICMP diferente das anteriores, o programa *traceroute* sabe que o destino final foi alcançado e usa as informações presentes na mensagem recebida para determinar o endereço IP e o nome do destino final.

O programa *traceroute* também define um temporizador para determinar o RTT até cada roteador e até o destino. A maioria dos programas *traceroute* envia três mensagens para cada dispositivo, com o mesmo valor de TTL, com o objetivo de estimar melhor o valor do RTT. A seguir, mostramos um exemplo de um programa *traceroute* que envia três mensagens para cada dispositivo e recebe três RTTs como resposta.

```
$ traceroute impressoras.com
traceroute para impressoras.com (13.1.69.93), 30 saltos max, 38 bytes por pacote

1 route.front.edu         (153.18.31.254)    0.622 ms    0.891 ms    0.875 ms
2 ceneric.net             (137.164.32.140)   3.069 ms    2.875 ms    2.930 ms
3 satire.net              (132.16.132.20)    3.071 ms    2.876 ms    2.929 ms
4 alpha.impressoras.com   (13.1.69.93)       5.922 ms    5.048 ms    4.922 ms
```

Nesse caso, há três saltos entre a origem e o destino. Observe que alguns RTTs parecem estranhos. É possível que um roteador esteja ocupado demais para processar o pacote imediatamente.

4.3 ROTEAMENTO *UNICAST*

Em uma internet, o objetivo da camada de rede é entregar um datagrama da sua origem ao seu destino ou destinos. Se um datagrama deve ser entregue a apenas um destino (entrega um para um), temos o chamado *roteamento unicast*. Se o datagrama deve ser entregue a vários destinos (entrega um para muitos), temos o chamado *roteamento multicast*. Finalmente, se o datagrama deve ser entregue a todas as estações da internet (um para todos), temos o chamado *roteamento broadcast*. Nesta seção e na próxima, discutiremos apenas o roteamento *unicast*; os roteamentos *multicast* e *broadcast* serão discutidos mais adiante.

O roteamento *unicast* na Internet, na qual existe um grande número de roteadores e de estações, só pode ser realizado usando roteamento hierárquico: em várias etapas, utilizando diferentes algoritmos de roteamento. Nesta seção, primeiro discutimos o conceito geral de roteamento *unicast* em uma *internet*: uma rede de redes cujas interconexões são feitas por meio de roteadores. Depois que os conceitos e algoritmos de roteamento tiverem sido compreendidos, mostraremos como podemos aplicá-los à Internet usando roteamento hierárquico.

4.3.1 Ideia geral

No roteamento *unicast*, um pacote é roteado, salto a salto, da origem até o destino com a ajuda de tabelas de roteamento. A estação de origem não precisa de uma tabela de roteamento porque ela entrega seu pacote para o roteador-padrão em sua rede local. A estação de destino também não

precisa de uma tabela de roteamento porque ela recebe o pacote do seu roteador-padrão em sua rede local. Isso significa que apenas os roteadores que interligam as redes na internet precisam de tabelas de roteamento. Dada a explicação anterior, o roteamento de um pacote da origem até o destino corresponde ao roteamento do pacote de um *roteador de origem* (o roteador-padrão da estação de origem) até um *roteador de destino* (o roteador conectado à rede de destino). Apesar de um pacote precisar necessariamente passar pelos roteadores de origem e de destino, a questão é: por quais outros roteadores o pacote deve passar? Em outras palavras, existem várias rotas que um pacote pode tomar da origem até o destino; o que deve ser determinado é qual rota o pacote deve tomar.

Modelando uma internet como um grafo

Na busca pelo melhor caminho, uma internet pode ser modelada como um *grafo*. Um grafo em ciência da computação refere-se a um conjunto de *nós* e *arestas* (linhas) que conectam os nós. Para modelar uma internet como um grafo, podemos pensar em cada roteador como um nó e em cada conexão entre um par de roteadores como uma aresta. Uma internet é, na realidade, modelada como um *grafo ponderado*, no qual um custo é associado a cada aresta. Se um grafo ponderado for utilizado para representar uma área geográfica, os nós poderiam ser cidades, e as arestas as estradas ligando as cidades; os pesos, nesse caso, poderiam ser as distâncias entre cidades. No roteamento, entretanto, o custo de uma aresta tem uma interpretação diferente para protocolos de roteamento distintos, algo que discutiremos em uma seção mais adiante. Por ora, assumimos que existe um custo associado a cada aresta. Se não houver arestas entre dois nós, o custo é considerado infinito. A Figura 4.56 mostra como uma internet pode ser modelada como um grafo.

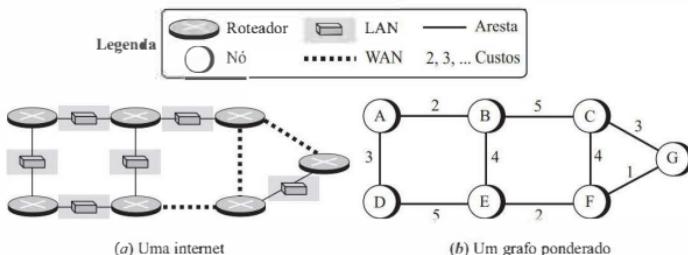



Figura 4.56 Uma internet e sua representação como um grafo.

Roteamento de menor custo

Quando uma internet é modelada como um grafo ponderado, uma das maneiras de interpretar a *melhor* rota do roteador de origem até o roteador de destino é considerar que ela corresponde à rota com *menor custo* entre os dois. Em outras palavras, o roteador de origem escolhe a rota até o roteador de destino de tal maneira que o custo total do percurso seja o menor possível dentre todas as rotas possíveis. Na Figura 4.56, a melhor rota entre A e E é A-B-E, cujo custo é 6. Isto significa que cada roteador precisa encontrar a rota de menor custo entre ele mesmo e todos os outros roteadores para ser capaz de rotear um pacote usando tais critérios.

Árvores de menor custo

Se houver N roteadores em uma internet, existem $(N - 1)$ caminhos de menor custo partindo de cada roteador para qualquer outro roteador. Isto significa que precisamos de $N \times (N - 1)$ caminhos de menor custo para a internet toda. Se existem apenas 10 roteadores em uma internet, precisamos de

90 caminhos de menor custo. Uma forma melhor de ver todos esses caminhos é combinando-os em uma **árvore de menor custo**, uma árvore que tem o roteador de origem como raiz e que se estende por todo o grafo (visita todos os outros nós) e na qual o caminho entre a raiz e qualquer outro nó é o mais curto. Dessa forma, pode haver apenas uma árvore de menor custo para cada nó e, portanto, temos *N* árvores de menor custo para toda a internet. Mostraremos como criar uma árvore de menor custo para cada nó mais adiante nesta seção; por ora, a Figura 4.57 mostra as sete árvores de menor custo para a internet da Figura 4.56.

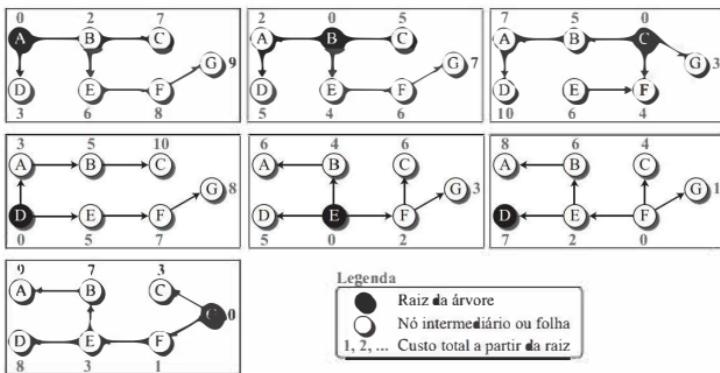



Figura 4.57 Árvores de menor custo para os nós na internet da Figura 4.56.

As árvores de menor custo para um grafo ponderado podem apresentar diversas propriedades se elas forem criadas usando critérios consistentes.

1. A rota de menor custo de X para Y na árvore de X é equivalente ao inverso da rota de menor custo de X para Y na árvore de Y; o custo em ambas as direções é o mesmo. Por exemplo, na Figura 4.57, a rota de A para F na árvore de A é (A → B → E → F), enquanto a rota de F para A na árvore de F é (F → E → B → A), que é o inverso da primeira rota. O custo é 8 em ambos os casos.
2. Em vez de ir de X até Z usando a árvore de X, podemos ir de X até Y usando a árvore de X e continuar a viagem de Y até Z usando a árvore de Y. Por exemplo, na Figura 4.57, podemos ir de A até G usando a árvore de A por meio da rota (A → B → E → F → G). Por outro lado, podemos ir de A para E usando a árvore de A (A → B → E) e então continuar o caminho usando a árvore de E por meio da rota (E → F → G). A combinação das duas rotas no segundo caso é equivalente à rota do primeiro caso. O custo no primeiro caso é 9 e o custo no segundo caso também é 9 (6 + 3).

4.3.2 Algoritmos de roteamento

Agora que discutimos a ideia geral por trás das árvores de menor custo e sobre as tabelas de roteamento que podem ser criadas a partir delas, nós nos concentraremos nos algoritmos de roteamento. Diversos algoritmos de roteamento foram projetados no passado. As diferenças entre os métodos concentram-se na forma como eles interpretam o menor custo e no modo como criam as árvores

de menor custo para cada nó. Nesta seção, discutimos os algoritmos mais comuns; mais adiante, mostraremos como um protocolo de roteamento na Internet implementa um desses algoritmos.

Roteamento por vetor de distâncias

O roteamento por **Vetor de Distâncias** (DV – Distance Vector) tem como objetivo a minimização de custos na escolha da melhor rota. No roteamento por vetor de distâncias, a primeira coisa que cada nó cria é a sua própria árvore de menor custo contendo as informações básicas que aquele nó possui sobre os seus vizinhos imediatos. As árvores incompletas são trocadas entre vizinhos imediatos para fazer com que as árvores fiquem cada vez mais completas, de modo a representar a internet inteira. Podemos dizer que no roteamento por vetor de distâncias, um roteador continuamente informa todos os seus vizinhos sobre aquilo que ele conhece sobre a rede (embora tal conhecimento possa ser incompleto).

Antes de mostrar como árvores de menor custo incompletas podem ser combinadas para criar árvores mais completas, precisamos discutir dois pontos importantes: a equação de Bellman-Ford e o conceito de vetores de distância, que cobrimos a seguir.

Equação de Bellman-Ford

O coração de roteamento por vetor de distâncias é a famosa equação de **Bellman-Ford**. Essa equação é usada para determinar o menor custo (distância mais curta) entre um nó de origem ***x***, e um nó de destino ***y***, através de alguns nós intermediários (a, b, c, ...) dados (1) os custos entre a origem e os nós intermediários e (2) os menores custos entre os nós intermediários e o destino. A seguir, mostraremos o caso geral em que D_{ij} é a distância mais curta e c_{ij} é o custo entre os nós ***i*** e ***j***.

$$D_{xy} = \min \{(c_{xa} + D_{ay}), (c_{xb} + D_{by}), (c_{xc} + D_{cy}), \ldots\}$$

No roteamento por vetor de distâncias, normalmente queremos atualizar um custo mínimo existente com outro custo mínimo através de um nó intermediário, como *z*, caso este seja menor. Nesse caso, a equação fica mais simples, conforme mostrado a seguir:

$$D_{xy} = \min \{D_{xy}, (c_{xz} + D_{zy})\}$$

A Figura 4.58 ilustra graficamente a ideia para ambos os casos.

(*a*) Caso geral com três nós intermediários (*b*) Atualizando um caminho com uma nova rota

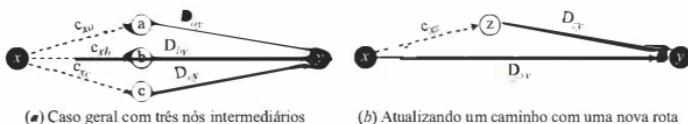



Figura 4.58 Ideia por trás da equação de Bellman-Ford.

Podemos dizer que a equação de Bellman-Ford nos permite construir um novo caminho de menor custo a partir de caminhos de menor custo previamente estabelecidos. Na Figura 4.58, podemos pensar em $(a \rightarrow y)$, $(b \rightarrow y)$ e $(c \rightarrow y)$ como caminhos de menor custo previamente estabelecidos e em $(x \rightarrow y)$ como o novo caminho de menor custo. Podemos até mesmo pensar

nessa equação como a construtora de uma nova árvore de menor custo a partir de árvores de menor custo previamente estabelecidas se aplicarmos essa equação repetidamente. Em outras palavras, o uso dessa equação no roteamento por vetor de distâncias é uma evidência de que o método também utiliza árvores de menor custo, mas essa utilização pode não ser facilmente perceptível.

Mostraremos, em breve, como podemos usar a equação de Bellman-Ford e o conceito de vetores da distância para construir caminhos de menor custo para cada nó no roteamento por vetor de distâncias. Entretanto, primeiro é preciso discutir o conceito de vetor de distâncias.

Vetores de distâncias

O conceito de **vetor de distâncias** é o que dá origem ao nome do *roteamento por vetor de distâncias*. Uma árvore de menor custo é uma combinação de caminhos de menor custo da raiz da árvore até todos os destinos. Esses caminhos são combinados graficamente para formar a árvore. O roteamento por vetor de distâncias separa os caminhos e cria um *vetor de distâncias*, um vetor unidimensional que representa a árvore. A Figura 4.59 mostra a árvore para o nó A na internet da Figura 4.56 e seu correspondente vetor de distâncias.

(*a*) Árvore para o nó A (*b*) Vetor de distâncias para o nó A

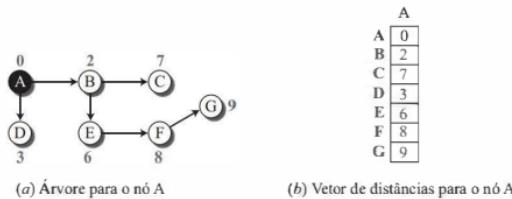



Figura 4.59 O vetor de distâncias correspondente a uma árvore.

Perceba que o *nome* do vetor de distâncias define a raiz, os *índices* definem os destinos e o *valor* de cada célula define o menor custo da raiz até o destino. Um vetor de distâncias não fornece o caminho para os destinos como faz a árvore de menor custo; ele fornece apenas os custos mínimos até os destinos. Mais à frente, mostraremos como podemos transformar um vetor de distâncias em uma tabela de roteamento, mas antes precisamos determinar todos os vetores de distâncias de uma internet.

Sabemos que um vetor de distâncias pode representar caminhos de menor custo em uma árvore de menor custo, mas a questão é: como cada nó em uma internet originalmente cria o vetor correspondente? Quando inicializado, cada nó em uma internet cria um vetor de distâncias muito rudimentar, contendo o conjunto mínimo de informações que o nó é capaz de obter de sua vizinhança. O nó envia algumas mensagens de saudação por suas interfaces e descobre a identidade dos vizinhos imediatos e a distância entre ele próprio e cada vizinho. Em seguida, cria um vetor de distâncias simples, inserindo as distâncias descobertas nas células correspondentes, e preenche o valor das outras células com infinito. Será que esses vetores de distâncias representam os caminhos de menor custo? A resposta é sim, considerando a pouca quantidade de informações que o nó possui. Quando conhecemos apenas uma distância entre dois nós, essa é a distância de menor custo. A Figura 4.60 mostra todos os vetores de distâncias para a nossa internet. No entanto, é preciso mencionar que esses vetores são criados de forma assíncrona, quando o nó correspondente é inicializado; o fato de todos eles estarem presentes na figura não significa que tenham sido criados ao mesmo tempo.

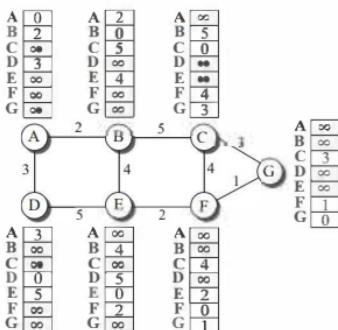



Figura 4.60 O vetor de distâncias inicial em uma internet.

Os vetores rudimentares não são capazes de ajudar a internet a efetivamente encaminhar um pacote. Por exemplo, o nó A acredita que não está ligado ao nó G porque a célula correspondente mostra que o menor custo entre eles é infinito. Para melhorar esses vetores, os nós da internet precisam ajudar uns aos outros, trocando informações. Depois de criar seu vetor, cada nó envia uma cópia deste vetor para todos os seus vizinhos imediatos. Após receber um vetor de distâncias de um vizinho, o nó atualiza seu vetor de distâncias usando a equação de Bellman-Ford (segundo caso). No entanto, é importante perceber que precisamos atualizar não apenas um menor custo, mas *N* deles, onde *N* é o número de nós na internet. Quando utilizamos um programa de computador, podemos fazer isso usando um laço; se estivermos ilustrando o conceito no papel, podemos mostrar o vetor inteiro em vez das *N* equações separadas. Na Figura 4.61, mostramos o vetor inteiro em vez de sete equações para cada atualização. A figura mostra dois eventos assíncronos, acontecendo um após o outro com algum intervalo entre eles. No primeiro evento, o nó A envia seu vetor para o nó B. O nó B atualiza seu vetor usando o custo $c_{BA} = 2$. No segundo evento, o nó E envia seu vetor para o nó B. O nó B atualiza seu vetor usando o custo $c_{EA} = 4$.

	Novo B	Antigo B	A
A	2	2	0
B	0	0	2
C	5	5	∞
D	**5**	∞	3
E	4	4	∞
F	∞	∞	∞
G	∞	∞	∞

B[ ] = min (B[ ], 2 + A[ ])

	Novo B	Antigo B	E
A	2	2	∞
B	0	0	4
C	5	5	∞
D	5	5	5
E	4	4	0
F	**6**	∞	2
G	∞	∞	∞

B[ ] = min (B[ ], 4 + E[ ])

(*a*) Primeiro evento: B recebe uma cópia do vetor A. (*b*) Segundo evento: B recebe uma cópia do vetor E.

Nota:
X[ ]: o vetor completo

Figura 4.61 Atualizando vetores de distâncias.

Após o primeiro evento, o vetor do nó B apresenta uma melhoria: o menor custo para o nó D mudou de infinito para 5 (via nó A). Após o segundo evento, o vetor do nó B apresenta mais uma melhoria: o seu menor custo para o nó F mudou de infinito para 6 (via nó E). Esperamos ter convencido o leitor de que a troca de vetores estabiliza o sistema e permite que todos os nós determinem o menor custo final entre eles e qualquer outro nó. Precisamos lembrar que, após a atualização de um nó, este envia imediatamente seu vetor atualizado para todos os seus vizinhos. Mesmo que seus vizinhos tenham recebido o vetor anterior, a nova atualização pode ajudar ainda mais.

Algoritmo de roteamento por vetor de distâncias

Agora, temos condições de estudar um pseudocódigo simplificado para o algoritmo de roteamento por vetor de distâncias, conforme mostrado na Tabela 4.4. O algoritmo é executado pelo seu nó de forma independente e assíncrona.

Tabela 4.4 Algoritmo de roteamento por vetor de distâncias para o nó A.

```
1   Roteamento_Vetor_de_Distancias ( )
2   {
3       // Inicialização (cria vetor inicial para o nó)
4       D[eu_mesmo] = 0
5       para (y = 1 até N)
6       {
7           se (y é um vizinho)
8               D[y] = c[eu_mesmo][y]
9           senão
10              D[y] = ∞
11      }
12      envia vetor {D[1], D[2], ..., D[N]} para todos os vizinhos
13      // Atualização (melhora o próprio vetor com o vetor recebido de um vizinho)
14      repetir (sempre)
15      {
16      espere (para um vetor Dw vindo de um vizinho w ou qualquer alteração no enlace)
17      para (y = 1 até N)
18      {
19          D[y] = min [D[y], (c[eu_mesmo][w] + Dw[y])]          // Equação de Bellman-Ford
20      }
21      se (qualquer alteração no vetor)
22          envia vetor {D[1], D[2], ..., D[N]} a todos os vizinhos
23      }
24  } // Fim do Vetor de Distâncias
```

As linhas 4 a 11 inicializam o vetor do nó. As linhas 14-23 mostram como o vetor pode ser atualizado depois da recepção de um vetor vindo do vizinho mais próximo. O laço ***para*** nas linhas 17 a 20 permite que todas as entradas (células) no vetor sejam atualizadas depois da recepção de um novo vetor. Observe que o nó envia seu vetor na linha 12, após sua inicialização, e na linha 22, após sua atualização.

Contagem até o infinito

Um problema com roteamento por vetor de distâncias é que qualquer redução no custo (boa notícia) propaga-se rapidamente, mas qualquer aumento no custo (má notícia) se propaga lentamente. Para que um protocolo de roteamento funcione corretamente, se uma conexão física falhar (custo torna-se infinito), cada outro roteador deve tomar ciência desse fato imediatamente; porém, no roteamento por vetor de distâncias, isso leva algum tempo. Esse problema é conhecido como *contagem até o infinito*. Algumas vezes, são necessárias diversas atualizações para que o custo de uma conexão quebrada seja registrado como o infinito por todos os roteadores.

Laço infinito entre dois nós

Um exemplo de contagem até o infinito é o chamado problema do laço infinito entre dois nós (*two-node loop problem*). Para facilitar a compreensão desse problema, analisaremos o cenário ilustrado na Figura 4.62.

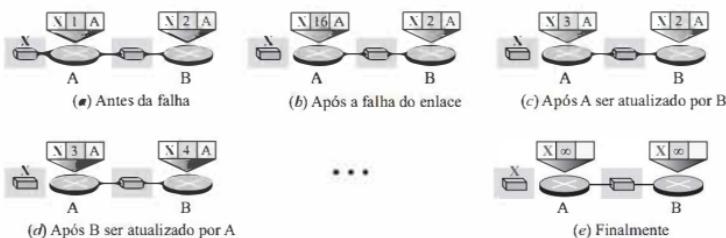
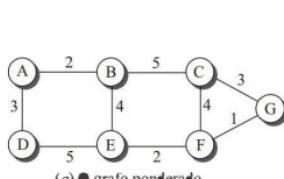



Figura 4.62 Instabilidade entre dois nós.

A figura mostra um sistema com três nós. Mostramos apenas as partes da tabela de roteamento necessárias para nossa análise. No início, tanto o nó A como o nó B sabem como chegar ao nó X, porém, de repente, a conexão física entre A e X falha. O nó A altera sua tabela. Se A for capaz de enviar sua tabela para B imediatamente, tudo termina bem, mas o sistema torna-se instável se B enviar sua tabela de roteamento para A antes de receber a tabela de roteamento de A. Nesse último caso, o nó A recebe a atualização de B e, acreditando que B tenha encontrado uma maneira de chegar a X, imediatamente atualiza sua tabela de roteamento. Agora, A envia sua nova atualização para B. O nó B pensa, então, que algo foi alterado na vizinhança de A e, portanto, atualiza sua tabela de roteamento. O custo para chegar até X aumenta gradualmente até atingir o valor infinito. Nesse momento, tanto A como B sabem que X não pode ser alcançado. No entanto, durante todo o tempo, o sistema fica instável. O nó A acredita que a rota até X é via B; já o nó B acredita que a rota para X é via A. Se A receber um pacote destinado a X, o pacote será encaminhado para B e depois voltará para A. Da mesma forma, se B receber um pacote destinado a X, tal pacote será encaminhado para A e de volta para B. Os pacotes ficam indo e voltando entre A e B, criando um laço infinito entre os dois nós. Algumas soluções foram propostas para instabilidades desse tipo.

Horizonte dividido Uma solução para instabilidades é o denominado *horizonte dividido* (*split horizon*). Nela, em vez de enviar sua tabela por todas as interfaces, cada nó envia apenas parte de sua tabela por cada interface. Se, de acordo com sua tabela, o nó B acredita que a melhor rota para chegar até X é via A, não é necessário anunciar essa informação para A, já que ela veio de A (ou seja, A já sabe disso). O fato de pegar informações do nó A, modificá-las e enviá-las de volta para o nó A é o que cria confusão. No nosso cenário, o nó B precisa eliminar a última linha de sua tabela de roteamento antes de enviá-la para A. Nesse caso, o nó A mantém o valor de infinito como a distância até X. Mais tarde, quando o nó A enviar sua tabela de roteamento para B, o nó B também corrige sua tabela de roteamento com as informações fornecidas por A. O sistema se estabiliza após a primeira atualização: tanto o nó A como o nó B sabem que X não pode ser alcançado.

Envenenamento reverso O uso da estratégia horizonte dividido tem uma desvantagem. Normalmente, o protocolo correspondente usa um temporizador e, caso não receba notícias sobre uma rota, o nó exclui a rota de sua tabela. Quando o nó B no cenário anterior eliminar a rota para X do seu anúncio para o nó A, este é incapaz de determinar se isso aconteceu devido à estratégia de horizonte dividido (a origem das informações foi A) ou porque B não recebeu qualquer notícia sobre X recentemente. Na estratégia de envenenamento reverso (*Poisoned Reverse*), B ainda pode anunciar o valor de seu custo até X para o nó A, mas se a fonte de tal informação for A, o nó B substitui a distância conhecida pelo valor infinito. Isto funciona como um aviso: "Não use esse valor, pois o que eu sei sobre essa rota veio de você."

Instabilidade entre três nós

A instabilidade entre dois nós pode ser evitada pelo uso do horizonte dividido em combinação com o envenenamento reverso. Entretanto, se uma instabilidade envolver três nós, a estabilidade não pode ser garantida.

Roteamento por estado de enlace

Um algoritmo de roteamento que segue diretamente nossa discussão para a criação de árvores e tabelas de roteamento de custo mínimo é o **roteamento por estado de enlace** (LS – link-state). Esse método usa o termo *estado de enlace* para definir as características de um enlace (uma aresta) na internet. Nele, o custo associado a uma aresta define o estado do enlace. Enlaces com custos mais baixos são preferíveis a enlaces com custos mais elevados; se o custo de um enlace for infinito, significa que o enlace não existe ou está quebrado.

Base de dados de estado de enlace

Para criar uma árvore de menor custo com este método, cada nó precisa ter um ***mapa*** completo da rede, ou seja, ele precisa saber o estado de cada enlace. O conjunto de estados de todos os enlaces é denominado **Base de Dados de Estado de Enlace** (LSDB – Link-State DataBase). Existe apenas uma LSDB para a internet inteira; cada nó precisa ter uma réplica dela para ser capaz de criar a árvore de menor custo. A Figura 4.63 mostra um exemplo de uma LSDB para o grafo da Figura 4.56. A LSDB pode ser representada como um vetor bidimensional (matriz), no qual o valor de cada célula define o custo do enlace correspondente.

(*a*) O grafo ponderado

	A	B	C	D	E	F	G
A	0	2	∞	3	∞	∞	∞
B	2	0	5	∞	4	∞	∞
C	∞	5	0	∞	∞	4	3
D	3	∞	∞	0	5	∞	∞
E	∞	4	∞	5	0	2	∞
F	∞	∞	4	∞	2	0	1
G	∞	∞	3	∞	∞	1	0

(*b*) Base de dados de estado de enlace

Figura 4.63 Exemplo de uma base de dados de estado de enlace.

Neste caso, a questão é como cada nó pode criar a LSDB contendo informações sobre toda a internet, o que pode ser feito por meio de um processo de **inundação**. Cada nó pode enviar algumas mensagens de saudação a todos os seus vizinhos imediatos (os nós aos quais ele está conectado diretamente) com o objetivo de coletar duas informações de cada nó vizinho: a identidade do nó e o custo do enlace. O pacote que contém essas duas informações é denominado *pacote LS* (LSP – Link-State Packet); o LSP é enviado por todas as interfaces, conforme mostra a Figura 4.64, para a internet da Figura 4.56. Quando um nó recebe um LSP em uma de suas interfaces, ele compara o LSP com a cópia que já possui. Se o LSP recém-obtido for mais antigo do que o atual (algo determinado pela verificação do número de sequência), este LSP é descartado. Se for mais recente ou o primeiro recebido, o nó descarta seu antigo LSP (caso exista) e conserva o LSP recém-recebido. Em seguida, envia uma cópia do mesmo LSP por cada uma de suas interfaces, exceto por aquela na qual o pacote chegou. Essa estratégia garante que as inundações terão fim em algum lugar na rede (quando um nó tiver uma única interface). Precisamos nos convencer de que, depois de receber todos os novos LSPs, cada nó será capaz de criar a LSDB completa, conforme mostra a Figura 4.64. Essa LSDB é idêntica para cada nó e fornece o mapa da internet inteira. Em outras palavras, um nó pode utilizar a LSDB para criar o mapa completo, caso necessário.

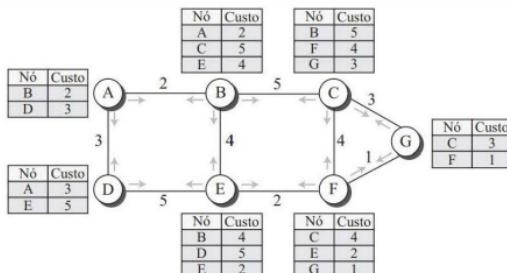



Figura 4.64 LSPs criados e enviados por cada nó para a construção da LSDB.

É possível comparar o algoritmo de roteamento por estado de enlace com o algoritmo de roteamento por vetor de distâncias. Neste, cada roteador informa a seus vizinhos aquilo que sabe sobre toda a internet; no algoritmo de roteamento por estado de enlace, cada roteador informa à internet tudo aquilo que ele sabe sobre seus vizinhos.

Formação de árvores de menor custo

Para criar uma árvore de menor custo para si próprio usando a LSDB compartilhada, cada nó precisa executar o famoso **Algoritmo de Dijkstra**. Este algoritmo iterativo consiste nos seguintes passos:

1. O nó escolhe a si mesmo como a raiz da árvore, criando uma árvore com um único nó, e fixa o custo total de cada nó com base nas informações da LSDB.
2. O nó seleciona um nó, dentre todos aqueles que não estão na árvore, que seja o mais próximo da raiz, e então o adiciona à árvore. Após a adição, os custos de todos os outros nós que não estão na árvore precisam ser atualizados, porque os caminhos podem ter sido alterados.
3. O nó repete o passo 2 até que todos os nós sejam adicionados à árvore.

Precisamos nos convencer de que os três passos anteriores finalmente criam a árvore de menor custo. A Tabela 4.5 mostra uma versão simplificada do Algoritmo de Dijkstra.

Tabela 4.5 Algoritmo de Dijkstra.

1	**Algoritmo_de_Dijkstra ( )**	
2	{	
3	// Inicialização	
4	Arvore = {raiz}	// A árvore é composta apenas pela raiz
5	**para** (*y* = 1 até N)	// N corresponde ao número de nós
6	{	
7	**se** (*y* é a raiz)	
8	D[*y*] = 0	// D[y] é a menor distância da raiz até o nó
9	**senão se** (y é um vizinho)	
10	D[*y*] = c[raiz][y]	// c[x][y] é o custo entre os nós x e y na LSDB
11	**senão**	
12	D[*y*] = ¥	
13	}	
14	// Cálculo	
15	**repetir**	
16	{	
17	encontre um nó w de modo que D[w] seja o mínimo dentre todos os nós que não pertencem à Árvore	
18	Árvore = Árvore É {w}	// Adiciona w à árvore
19	// Atualiza as distâncias para todos os vizinhos de w	
20	**para** (todos os nós *x* que sejam vizinhos de w e não pertençam à Árvore)	
21	{	
22	D[*x*] = min{D[*x*], (D[*w*] + c[*w*][*x*])}	
23	}	
24	} **até** (todos os nós pertencem à Árvore)	
25	} // Fim do algoritmo de Dijkstra	

As linhas 4 a 13 implementam o passo 1 do algoritmo. As linhas 16 a 23 implementam o passo 2 do algoritmo. O passo 2 é repetido até que todos os nós sejam adicionados à árvore.

A Figura 4.65 mostra a formação da árvore de menor custo para o grafo da Figura 4.63, usando o algoritmo de Dijkstra. É preciso passar pela inicialização e por seis iterações para determinar a árvore de menor custo.

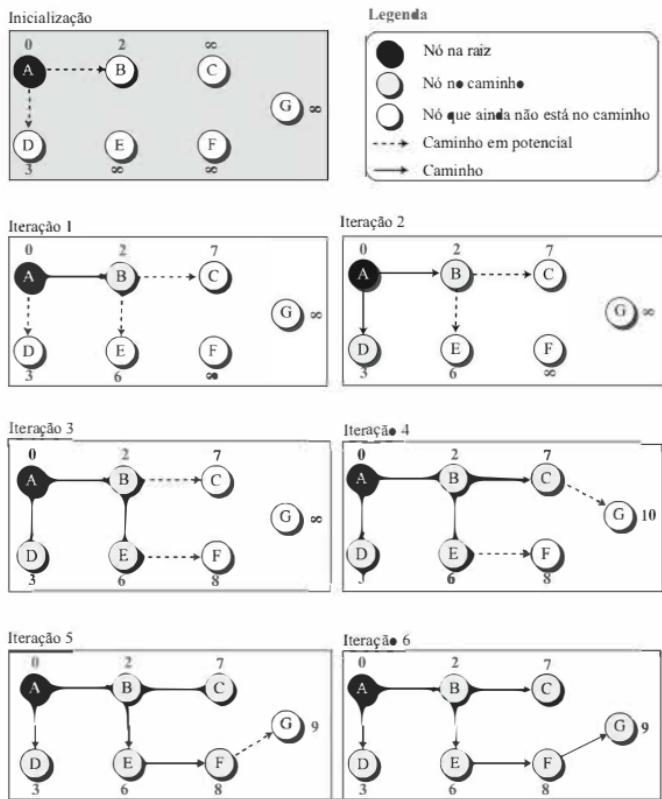



Figura 4.65 Árvore de menor custo.

Roteamento por vetor de caminhos

Tanto o roteamento por estado de enlace como por vetor de distâncias são baseados na meta de menor custo. No entanto, existem casos em que tal meta não é a prioridade. Por exemplo, considere que existam alguns roteadores na internet pelos quais um remetente quer evitar que seus pacotes passem. É possível que esses roteadores pertençam a uma organização que não oferece segurança suficiente, ou que eles pertençam a um rival comercial do remetente que poderia inspecionar os pacotes em busca de informações. O roteamento de menor custo não impede que um pacote passe por uma área da rede quando esta for parte do caminho de menor custo. Em outras palavras, o objetivo do menor custo, adotado no roteamento por LS ou por DV, não permite que um remetente aplique políticas específicas sobre a rota que um pacote pode tomar. Além de segurança e confiabilidade, existem ocasiões, conforme discutido na próxima seção, nas quais o objetivo do roteamento é

meramente a acessibilidade: permitir que o pacote chegue ao seu destino da forma mais eficiente, sem atribuição de custos à rota.

Para atender essas demandas, um terceiro algoritmo de roteamento, denominado **roteamento por vetor de caminhos** (PV – path-vector) foi concebido. O roteamento por vetor de caminhos não apresenta os inconvenientes dos roteamentos por LS ou por DV descritos anteriormente, porque ele não se baseia no roteamento de menor custo. A melhor rota é determinada pela origem usando a política que ela deseja impor sobre a rota, ou seja, a origem pode controlar o caminho dos pacotes. Embora o roteamento por vetor de caminhos não seja realmente usado em uma internet e seja projetado principalmente para rotear pacotes entre ISPs, discutimos os princípios desse método nesta seção como se ele fosse aplicado a uma internet. Na próxima seção, mostraremos como ele é usado na Internet.

Árvores de abrangência

No roteamento por vetor de caminhos, o caminho de uma origem até todos os destinos também é determinado pela melhor árvore de abrangência (*spanning tree*). A melhor árvore de abrangência, no entanto, não é a árvore de menor custo; é a árvore determinada pela origem quando impõe sua própria política. Se houver mais de uma rota para um destino, a origem pode escolher a rota que melhor atenda à sua política; uma origem pode aplicar diversas políticas ao mesmo tempo. Uma política comum é usar o número mínimo de nós a serem visitados (algo semelhante ao menor custo). Outra é evitar alguns nós como nós intermediários em uma rota.

A Figura 4.66 mostra uma pequena internet com apenas cinco nós. Cada origem criou a sua própria árvore de abrangência que atende a sua política. A política imposta por todas as fontes é usar o número mínimo de nós para chegar a um destino. A árvore de abrangência selecionada pelos nós A e E é tal que a comunicação não apresenta D como nó intermediário. De forma semelhante, a árvore de abrangência selecionada por B é tal que a comunicação apresenta C como nó intermediário.

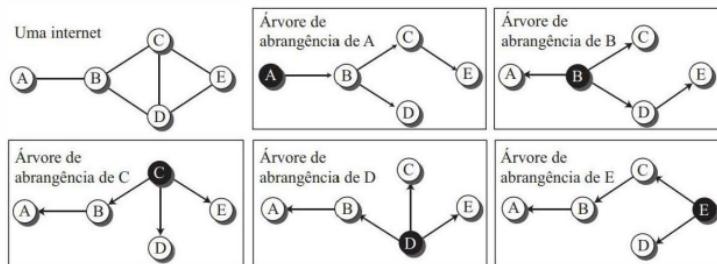



Figura 4.66 Árvores de abrangência (*spanning trees*) no roteamento de vetor de caminhos.

Criação de árvores de abrangência

O roteamento por vetor de caminhos, assim como o roteamento por vetor de distâncias, é um algoritmo de roteamento assíncrono e distribuído. As árvores de abrangência são criadas, gradual e assincronamente, por cada nó. Quando um nó é inicializado, ele cria um *vetor de caminhos* com base nas informações que obtém de seus vizinhos mais próximos. Um nó envia mensagens de saudação para seus vizinhos imediatos para coletar essas informações. A Figura 4.67 mostra todos esses vetores de caminhos para a internet da Figura 4.66. Perceba, entretanto, que não queremos dizer que todas essas tabelas são criadas simultaneamente, mas sim quando cada nó é inicializado. A figura também mostra como os vetores de caminhos são enviados para os vizinhos imediatos do nó após terem sido criados (setas).

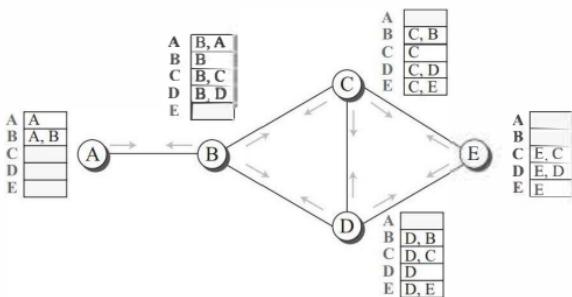
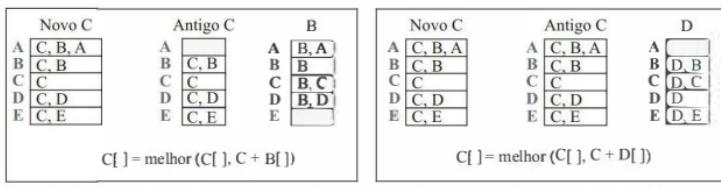



Figura 4.67 Vetores de caminhos criados no momento da inicialização.

Cada nó, após a criação do vetor de caminhos inicial, envia o vetor para todos os seus vizinhos imediatos. Cada nó, ao receber um vetor de caminhos de um vizinho, atualiza seu vetor de caminhos usando uma equação semelhante à de Bellman-Ford, mas aplica sua própria política em vez de procurar o menor custo. Podemos definir essa equação como

$$\text{Caminho}(x, y) = \text{melhor } \{\text{Caminho}(x, y), [(x + \text{Caminho}(v, y)]\} \text{ para todo } v \text{ na internet.}$$

Nessa equação, o operador (+) significa adicionar x ao início do caminho. Também precisamos ser cautelosos para evitar a adição de um nó a um caminho vazio, porque um caminho vazio equivale a um caminho inexistente.

A política é definida selecionando-se o *melhor* de diversos caminhos. O roteamento por vetor de caminhos também impõe uma condição a mais nessa equação: se Caminho (v, y) incluir x, ele é descartado para evitar um laço no caminho. Em outras palavras, x não deseja que ele mesmo seja visitado pelo pacote quando ele seleciona um caminho até y.

A Figura 4.68 mostra o vetor de caminhos do nó C após dois eventos. No primeiro evento, o nó C recebe uma cópia do vetor de B, melhorando seu próprio vetor: agora, ele sabe como chegar ao nó A. No segundo evento, o nó C recebe uma cópia do vetor de D, o que não muda seu próprio vetor. Na realidade, o vetor para o nó C após o primeiro evento é estabilizado e serve como sua tabela de roteamento.

Evento 1: C recebe uma cópia do vetor de B

	Novo C		Antigo C		D
A	C, B, A	A	C, B, A	A	
B	C, B	B	C, B	B	D, B
C	C	C	C	C	D, C
D	C, D	D	C, D	D	D
E	C, E	E	C, E	E	D, E

C[ ] = melhor (C[ ], C + D[ ])

Evento 2: C recebe uma cópia do vetor de D

Nota:
X [ ]: vetor X
Y: nó Y

Figura 4.68 Atualizando vetores de caminhos.

Algoritmo de vetor de caminhos

Com base no processo de inicialização e na equação utilizada para atualizar cada tabela de roteamento após o recebimento dos vetores de caminhos dos vizinhos, podemos escrever uma versão simplificada do algoritmo de vetor de caminhos, conforme mostrado na Tabela 4.6.

Tabela 4.6 Algoritmo de vetor de caminhos para um nó.

1	**Roteamento_Vetor_de_Caminhos ( )**
2	{
3	// Inicialização
4	**se** (*y* = 1 até N)
5	{
6	**se** (*y* = eu_mesmo)
7	Caminho[*y*] = eu_mesmo
8	**senão se** (*y* é um vizinho)
9	Caminho[*y*] = eu_mesmo + nó vizinho
10	**senão**
11	Caminho[*y*] = vazio
12	}
13	Envia vetor {Caminho[1], Caminho[2], ..., Caminho[y]} a todos os vizinhos
14	// Update
15	**repetir** (sempre)
16	{
17	**espere** (por um vetor $Caminho_w$ de um vizinho *w*)
18	**para** (*y* = 1 até N)
19	{
20	**se** ($Caminho_w$ inclui eu_mesmo)
21	descarte o caminho // Evita um laço
22	**senão**
23	Caminho[*y*] = **melhor** {Caminho[y], (eu_mesmo + $Caminho_w$[y])}
24	}
25	**se** (há alguma alteração no vetor)
26	**Envia** vetor {Caminho[1], Caminho[2], ..., Caminho[y]} para todos os vizinhos
27	}
28	} // Fim do Vetor de Caminhos

As linhas 4 a 12 mostram a inicialização do nó. As linhas 17 a 24 mostram como o nó atualiza seu vetor depois de receber um vetor de seu vizinho. O processo de atualização é repetido eternamente. Podemos observar as semelhanças entre esse algoritmo e o algoritmo de DV.

4.3.3 Protocolos de roteamento *unicast*

Na seção anterior, discutimos algoritmos de roteamento *unicast*; nesta, abordaremos protocolos de roteamento *unicast* utilizados na Internet. Embora os três protocolos aqui discutidos sejam baseados nos algoritmos correspondentes que discutimos anteriormente, um protocolo é mais que um algoritmo. Ele precisa definir seu domínio de operação, as mensagens trocadas, a comunicação entre roteadores e a interação com os protocolos em outros domínios. Após uma introdução, discutimos três protocolos comumente utilizados na Internet: o Protocolo de Informações de Roteamento (RIP – Routing Information Protocol), baseado no algoritmo de vetor de distâncias, o Protocolo Aberto de Menor Rota Primeiro (OSPF – Open Shortest Path First), baseado no algoritmo de estado de enlace e o Protocolo de Roteamento de Borda (BGP – Border Gateway Protocol), que é baseado no algoritmo de vetor de caminhos.

Estrutura da Internet

Antes de discutirmos os protocolos de roteamento *unicast*, precisamos entender a estrutura da Internet atual. A Internet transformou-se de uma estrutura em árvore, com um único *backbone*, na estrutura atual com múltiplos *backbones* gerenciados por diferentes empresas privadas. Embora seja difícil fornecer uma visão geral da Internet de hoje em dia, podemos dizer que ela tem uma estrutura semelhante à mostrada na Figura 4.69.

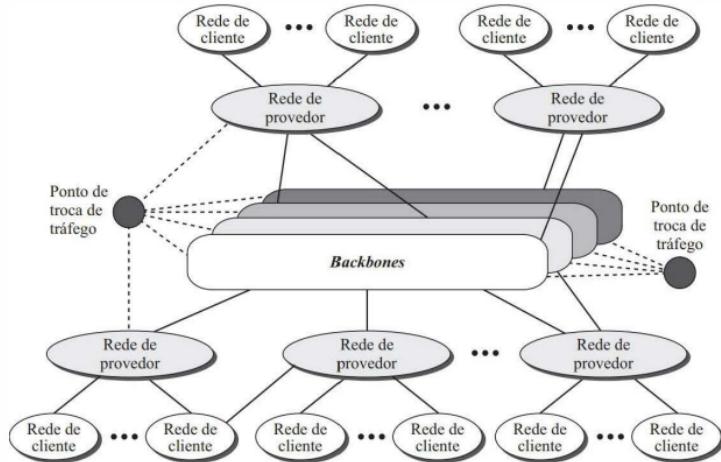



Figura 4.69 Estrutura da Internet.

Existem diversos *backbones* gerenciados por empresas privadas de comunicação que fornecem conectividade global. Esses *backbones* são conectados por alguns *pontos de troca de tráfego* (ou *peering points*) que permitem a conectividade entre os *backbones*. Em um nível mais abaixo, existem algumas *redes de provedores* que utilizam os *backbones* para conseguir conectividade global, mas prestam serviços a clientes da Internet. Finalmente, existem algumas *redes de clientes* que utilizam os serviços prestados pelas redes de provedores. Qualquer uma dessas três entidades

(*backbone*, rede de provedor ou rede de cliente) pode ser denominada um Provedor de Serviços de Internet (ISP – Internet Service Provider). Elas prestam serviços, mas em níveis diferentes.

Roteamento hierárquico

A Internet atual é composta por um grande número de redes e roteadores que as conectam. É óbvio que o roteamento na Internet não pode ser feito usando um único protocolo por dois motivos: um deles é a escalabilidade e outro é uma questão administrativa. O *problema da escalabilidade* significa que, como o tamanho das tabelas de roteamento seria muito grande, a busca por um destino tomaria muito tempo e as atualizações criariam um enorme volume de tráfego. A *questão administrativa* está relacionada à estrutura da Internet, descrita na Figura 4.69. Conforme mostrado nessa figura, cada ISP é gerenciado por uma autoridade administrativa. O administrador precisa ter controle sobre seu sistema. A organização deve ser capaz de usar tantas sub-redes e roteadores quanto forem necessários, pode desejar que os roteadores sejam todos de um fabricante em particular, pode querer executar um algoritmo de roteamento específico para atender às necessidades de sua organização e pode querer impor alguma política sobre o tráfego passando através de seu ISP.

Roteamento hierárquico significa considerar cada ISP como um **sistema autônomo** (AS – autonomous system). Cada AS pode executar um protocolo de roteamento que atenda as suas necessidades, mas a Internet global executa um protocolo global para interligar todos os AS. O protocolo de roteamento executado em cada AS é denominado um *protocolo de roteamento intra-AS*, *protocolo de roteamento intradomínio*, ou *Interior Gateway Protocol* (IGP); já o protocolo de roteamento global é denominado *protocolo de roteamento inter-AS*, *protocolo de roteamento interdomínios*, ou *Exterior Gateway Protocol* (EGP). Existem diversos protocolos de roteamento intradomínio, e cada AS é livre para escolher um deles, mas deve ficar claro que devemos ter apenas um protocolo interdomínios que trata o roteamento entre tais entidades. Atualmente, os dois protocolos de roteamento intradomínio mais comuns são o RIP e o OSPF; o único protocolo de roteamento interdomínios é o BGP. Essa situação pode mudar quando passarmos a usar o IPv6.

Sistemas autônomos

Como dissemos anteriormente, cada ISP é um sistema autônomo no que se refere à gestão das redes e roteadores sob seu controle. Embora existam AS de pequeno, médio e grande porte, cada AS recebe um número, denominado número de sistema autônomo (ASN – autonomous system number) pela ICANN. Cada ASN consiste em um número inteiro sem sinal de 16 *bits* que define univocamente um AS. Os sistemas autônomos, no entanto, não são classificados por tamanho, mas de acordo com a maneira como eles se conectam a outros AS. Existem AS *stub*, AS *multihomed* e AS de trânsito. O tipo, como veremos mais adiante, afeta a operação do protocolo de roteamento interdomínios com relação ao AS.

- **AS *stub*.** Um AS *stub* possui uma única conexão com outro AS. O tráfego de dados pode começar ou terminar em um AS *stub*; os dados não podem passar através dele. Um bom exemplo de um AS *stub* é uma rede de cliente, que é ora a origem e ora o destino dos dados.
- **AS *multihomed*.** Um AS *multihomed* pode ter mais de uma conexão a outros AS, mas não permite que o tráfego de dados passe através dele. Um bom exemplo desse tipo de AS é um AS cliente que utiliza os serviços de mais de uma rede de provedor, mas sua política não permite que dados sejam transferidos através de sua rede.
- **AS de trânsito.** Um AS de trânsito está conectado a mais de um AS e permite que o tráfego passe através dele. As redes de provedores e o *backbone* são bons exemplos de AS de trânsito.

Protocolo de Informações de Roteamento

O **Protocolo de Informações de Roteamento** (RIP – Routing Information Protocol) é um dos mais utilizados protocolos de roteamento intradomínio baseados no algoritmo de roteamento por vetor de distância, que descrevemos anteriormente. O RIP foi criado como parte do Sistema de Redes Xerox (XNS – Xerox Network System), mas foi a versão Distribuição de *Software* de Berkeley (BSD – Berkeley Software Distribution) do UNIX que ajudou a tornar o uso do RIP mais generalizado.

Contagem de saltos

Um roteador neste protocolo basicamente implementa o algoritmo de roteamento por vetor de distâncias mostrado na Tabela 4.4. Entretanto, o algoritmo foi modificado conforme descrito a seguir. Em primeiro lugar, como um roteador em um AS precisa saber como encaminhar um pacote para diferentes redes (sub-redes) em um AS, os roteadores RIP anunciam o custo para se alcançar diferentes redes em vez do custo para se alcançar outros nós em um grafo teórico. Em outras palavras, o custo é definido entre um roteador e a rede na qual a estação de destino está localizada. Em segundo lugar, para simplificar a implementação do custo (independentemente de fatores de desempenho dos roteadores e enlaces, como atraso, largura de banda e assim por diante), o custo é definido como o número de saltos, significando o número de sub-redes pelas quais um pacote precisa passar partindo do roteador de origem até chegar à estação de destino final. Perceba que a rede à qual a estação de origem está conectada não é contabilizada nesse cálculo, pois a estação de origem não usa uma tabela de roteamento; o pacote é entregue ao roteador-padrão. A Figura 4.70 ilustra o conceito de contagem de saltos anunciados por três roteadores desde uma estação de origem até uma estação de destino. No RIP, o custo máximo de um caminho vale 15, o que significa que 16 é tratado como infinito (nenhuma conexão). Por essa razão, o RIP só pode ser utilizado em sistemas autônomos nos quais o diâmetro do AS não seja superior a 15 saltos.

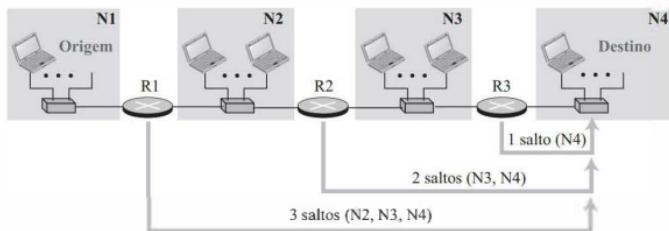



Figura 4.70 Contagem de saltos no RIP.

Tabelas de roteamento

Embora o algoritmo de vetor de distâncias discutido na seção anterior se preocupe com a troca de vetores de distâncias entre nós vizinhos, os roteadores em um sistema autônomo precisam manter tabelas de roteamento para encaminhar pacotes para as suas respectivas redes de destino. Uma tabela de roteamento no RIP é uma tabela com três colunas, na qual a primeira coluna é o endereço da rede de destino, a segunda é o endereço do próximo roteador para qual o pacote deve ser transmitido e a terceira é o custo (o número de saltos) para alcançar a rede de destino. A Figura 4.71 mostra as três tabelas de roteamento para os roteadores da Figura 4.70. Perceba que a primeira e a terceira colunas juntas fornecem as mesmas informações que um vetor de distâncias, mas o custo mostra o número de saltos até as redes de destino.

Tabela de roteamento de R1

Rede de destino	Próximo roteador	Custo em saltos
N1	——	1
N2	——	1
N3	R2	2
N4	R2	3

Tabela de roteamento de R2

Rede de destino	Próximo roteador	Custo em saltos
N1	R1	2
N2	——	1
N3	——	1
N4	R3	2

Tabela de roteamento de R3

Rede de destino	Próximo roteador	Custo em saltos
N1	R2	3
N2	R2	2
N3	——	1
N4	——	1

Figura 4.71 Tabelas de roteamento.

Apesar de uma tabela de roteamento RIP definir apenas o próximo roteador na sua segunda coluna, ela fornece informações sobre toda a árvore de menor custo com base na segunda propriedade dessas árvores, discutida na seção anterior. Por exemplo, R1 define que o próximo roteador no caminho até N4 é R2; R2 define que o próximo roteador para alcançar N4 é R3; R3 define que não há um próximo roteador para esse caminho. A árvore é, portanto, R1 → R2 → R3 → N4.

Uma pergunta frequente com relação à tabela de roteamento é sobre a utilidade da terceira coluna, desnecessária para a transmissão dos pacotes, mas necessária para atualizar a tabela de roteamento quando houver alguma mudança na rota, conforme veremos em breve.

Implementação do RIP

O RIP é implementado como um processo que utiliza os serviços do UDP na porta de número bem conhecido 520. No BSD, o RIP é um processo *daemon* (um processo que fica em execução em segundo plano) denominado *routed* (abreviação de *route daemon*). Isto significa que, embora o RIP seja um protocolo de roteamento que ajuda o IP a rotear seus datagramas através do AS, as mensagens RIP são encapsuladas em datagramas de usuário UDP, que, por sua vez, são encapsulados em datagramas IP. Ou seja, o RIP é executado na camada de aplicação, mas cria tabelas de roteamento para o IP na camada de rede.

O RIP passou por duas versões: RIP-1 e RIP-2. A segunda versão é retrocompatível com a primeira, o que permite o uso de mais informações nas mensagens RIP que tinham valor 0 na sua primeira versão. Discutimos apenas o RIP-2 nesta seção.

Mensagens RIP Dois processos RIP, um cliente e um servidor, como quaisquer outros processos, precisam trocar mensagens. O RIP-2 define o formato das mensagens, conforme mostra a Figura 4.72. Parte da mensagem, que chamamos de entrada, pode ser repetida conforme necessário em uma mensagem. Cada entrada carrega as informações relacionadas a uma linha na tabela de roteamento do roteador que envia a mensagem.

Campos

Com: Comando. Pode ser um pedido (1) ou uma resposta (2)
Ver: Versão. A versão atual é 2
Família: Família de protocolo. Para o TCP/IP, o valor é 2
Identificação: Informação sobre o sistema autônomo
Endereço de rede: Endereço de destino
Máscara de sub-rede: Tamanho do prefixo
Endereço do próximo salto: Tamanho do endereço
Distância: Número de saltos até o destino

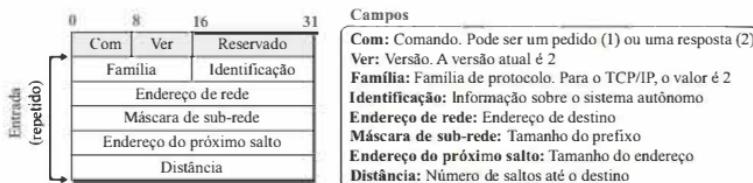



Figura 4.72 Formato das mensagens RIP.

O RIP apresenta dois tipos de mensagens: pedido e resposta. Uma mensagem de pedido é enviada por um roteador que acaba de entrar na rede ou por um roteador que tem algumas entradas desatualizadas. Uma mensagem de pedido pode perguntar sobre entradas específicas ou sobre todas as entradas. A mensagem de resposta (ou atualização) pode ou não ter sido solicitada. Uma mensagem de resposta solicitada é enviada apenas em resposta a uma mensagem de pedido. Ela contém informações sobre o destino especificado na mensagem de pedido correspondente. Uma mensagem de resposta não solicitada, por outro lado, é enviada periodicamente, a cada 30 segundos ou quando houver alguma mudança na tabela de roteamento.

Algoritmo RIP O RIP implementa o algoritmo de roteamento por vetor de distâncias discutido na seção anterior. No entanto, algumas mudanças precisam ser feitas no algoritmo para permitir que um roteador atualize sua tabela de roteamento:

- Em vez de enviar apenas vetores de distâncias, um roteador precisa enviar todo o conteúdo de sua tabela de roteamento em uma mensagem de resposta.
- O receptor adiciona um salto a cada custo e altera o campo de próximo roteador para o endereço do roteador que enviou a mensagem. Denominamos cada rota na tabela de roteamento modificada como *rota recebida* e cada rota na tabela de roteamento antigo como *antiga rota*. O roteador receptor conserva as rotas antigas como as novas rotas, exceto nos seguintes três casos:
  1. Se a rota recebida não existir na antiga tabela de roteamento, ela deve ser adicionada à tabela.
  2. Se o custo da rota recebida for menor do que o custo da antiga, a rota recebida deve ser escolhida como a nova rota.
  3. Se o custo da rota recebida for maior do que o custo da antiga, mas o valor do próximo roteador for o mesmo em ambas as rotas, a rota recebida deve ser escolhida como a nova rota. Este é o caso no qual a rota foi anunciada pelo mesmo roteador no passado, mas agora a situação se alterou. Por exemplo, suponha que um vizinho tenha anteriormente anunciado uma rota para um destino com custo 3, mas agora não haja um caminho entre o vizinho e o destino. O vizinho anuncia o destino com um custo de valor infinito (16 no RIP). O roteador que recebe o anúncio não deve ignorar esse valor, mesmo que sua antiga rota apresente um custo menor para o mesmo destino.
- A nova tabela de roteamento precisa ser ordenada de acordo com a rota de destino (geralmente colocando os prefixos mais longos no início).

Exemplo 4.15

A Figura 4.73 mostra um exemplo mais realista da operação do RIP em um sistema autônomo. Primeiro, a figura mostra todas as tabelas de roteamento após todos os roteadores terem sido inicializados. Em seguida, mostra as mudanças em algumas tabelas quando algumas mensagens de atualização são trocadas entre os roteadores. Finalmente, mostra as tabelas de roteamento estabilizadas quando não há mais mudanças.

Temporizadores no RIP O RIP usa três temporizadores para dar suporte à sua operação. O *temporizador periódico* controla o anúncio regular de mensagens de atualização. Cada roteador tem um temporizador periódico aleatoriamente inicializado com um número entre 25 e 35 segundos (para evitar que todos os roteadores enviem suas mensagens ao mesmo tempo, criando tráfego excessivo). O temporizador efetua uma contagem regressiva; quando chega a zero, a mensagem de atualização é enviada e o temporizador é novamente inicializado com um valor aleatório.

O *temporizador de expiração* controla a validade de uma rota. Quando um roteador recebe uma informação de atualização sobre uma rota, o temporizador de expiração é reinicializado com o valor de 180 segundos para aquela rota em particular. Toda vez que uma nova atualização para tal rota é recebida, o temporizador é reiniciado. Caso haja um problema na rede e nenhuma atualização

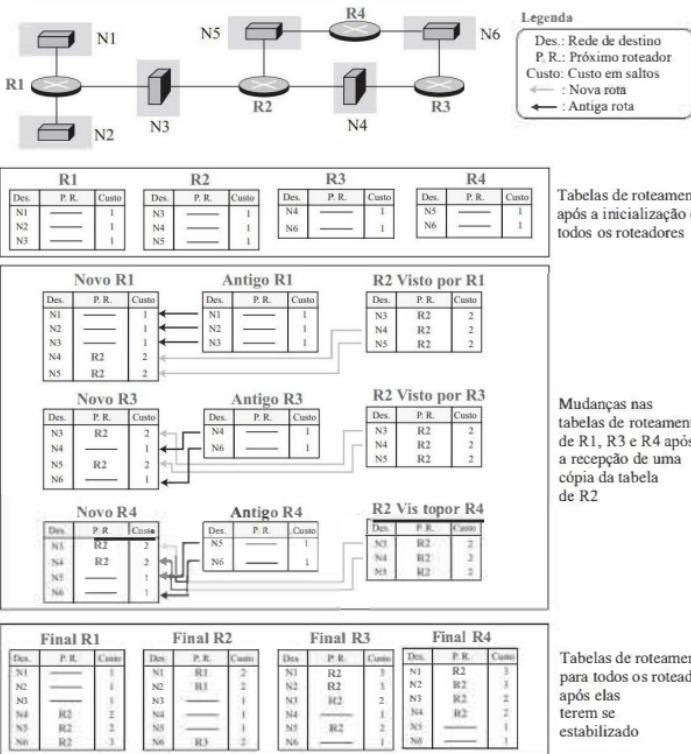



Figura 4.73 Exemplo de um sistema autônomo usando RIP.

seja recebida dentro dos 180 segundos especificados, a rota é considerada inválida e a contagem de saltos para aquela rota passa para 16, indicando que o destino está inacessível. Cada rota tem seu próprio temporizador de expiração. O *temporizador de coleta de lixo* (*garbage collection*) é usado para remover uma rota da tabela de roteamento. Quando a informação sobre uma rota se torna inválida, o roteador não remove imediatamente esse caminho de sua tabela, mas continua a anunciar a rota com um valor de 16. Ao mesmo tempo, um temporizador de coleta de lixo é inicializado em 120 segundos para essa rota. Quando a contagem regressiva desse temporizador chega a zero, a rota é removida da tabela. O temporizador permite que os vizinhos tomem ciência da invalidade de uma rota antes da sua remoção da tabela.

Desempenho

Antes de passarmos para a próxima seção, discutiremos brevemente questões de desempenho no RIP:

- **Mensagens de atualização.** As mensagens de atualização no RIP têm um formato muito simples e são enviadas somente para os vizinhos do roteador, sendo, portanto, mensagens locais. Elas normalmente não criam congestionamentos porque os roteadores tentam evitar que elas sejam enviadas ao mesmo tempo.
- **Convergência das tabelas de roteamento.** O RIP usa o algoritmo de vetor de distâncias, que pode convergir lentamente se o domínio for grande, mas como o RIP permite apenas 15 saltos em um domínio (16 são considerados infinitos), normalmente não ocorrem problemas de convergência. Os únicos problemas que podem retardar a convergência são situações de contagem até o infinito e os laços criados no domínio; a aplicação das estratégias de envenenamento reverso e horizonte dividido em conjunto com o RIP pode aliviar tais inconvenientes.
- **Robustez.** Como dissemos anteriormente, o roteamento por vetor de distâncias é baseado no conceito de que cada roteador envia o que ele sabe sobre o domínio inteiro para seus vizinhos. Isto significa que o cálculo da tabela de roteamento depende das informações recebidas de vizinhos imediatos, que por sua vez recebem informações de seus próprios vizinhos. Se um roteador falhar ou for corrompido, o problema vai se propagar para todos os roteadores e o processo de roteamento de cada roteador será afetado.

Protocolo Aberto de Menor Rota Primeiro

O **Protocolo Aberto de Menor Rota Primeiro** (OSPF – Open Shortest Path First) também é um protocolo de roteamento intradomínio como o RIP. Porém, o OSPF é baseado no protocolo de roteamento por estado de enlace que descrevemos anteriormente neste capítulo. O OSPF é um protocolo aberto (*open*), o que significa que sua especificação encontra-se em um documento de domínio público.

Métricas

No OSPF, tal como no RIP, o custo de se chegar a um destino partindo de uma estação de origem é calculado do roteador de origem até a rede de destino. No entanto, a cada enlace pode ser atribuído um peso com base na sua banda passante, no RTT correspondente, na sua confiabilidade e assim por diante. Um administrador da rede pode também decidir usar a contagem de saltos como custo. Um ponto interessante sobre o custo no OSPF é que diferentes tipos de serviço (TOS – Type of Service) podem incorrer em pesos diferentes nos custos. A Figura 4.74 mostra a ideia do custo de um roteador até a rede da estação de destino. Podemos comparar esta figura com a Figura 4.70, usada para o RIP.

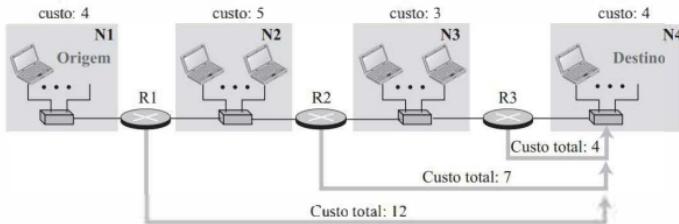



Figura 4.74 Métricas no OSPF.

Tabelas de roteamento

Cada roteador OSPF pode criar uma tabela de roteamento depois de encontrar a árvore de menor custo (ou árvore de caminho mais curto) entre ele próprio e o destino usando o algoritmo de

Dijkstra, descrito anteriormente neste capítulo. A Figura 4.75 mostra as tabelas de roteamento para o AS simples da Figura 4.74. Comparando as tabelas de roteamento para o OSPF e o RIP no mesmo AS, podemos perceber que a única diferença refere-se aos valores de custo. Em outras palavras, se usarmos a contagem de saltos no OSPF, as tabelas serão exatamente iguais. A razão para isso é que ambos os protocolos usam árvores de menor custo para determinar o melhor caminho de uma origem até um destino.

Tabela de roteamento de R1

Rede de destino	Próximo roteador	Custo
N1	——	4
N2	——	5
N3	R2	8
N4	R2	12

Tabela de roteamento de R2

Rede de destino	Próximo roteador	Custo
N1	R1	9
N2	——	5
N3	——	3
N4	R3	7

Tabela de roteamento de R3

Rede de destino	Próximo roteador	Custo
N1	R2	12
N2	R2	8
N3	——	3
N4	——	4

Figura 4.75 Tabelas de roteamento no OSPF.

Áreas

Em comparação com o RIP, que é normalmente utilizado em AS pequenos, o OSPF foi projetado para lidar com o roteamento em sistemas autônomos pequenos ou grandes. Contudo, a formação das árvores de menor custo no OSPF exige que todos os roteadores inundem o AS inteiro com os seus LSPs para criar a LSDB global. Embora isto possivelmente não crie problemas em um AS pequeno, tal mecanismo pode criar um enorme volume de tráfego em uma AS grande. Para que isso não ocorra, o AS precisa ser dividido em seções menores denominadas *áreas*. Cada área funciona como um pequeno domínio independente para inundação de LSPs. Em outras palavras, o OSPF utiliza um outro nível de hierarquia no roteamento: o primeiro nível é o sistema autônomo e o segundo é a área.

Entretanto, cada roteador em uma área precisa saber as informações sobre os estados de enlaces não apenas em sua área, mas também em outras. Assim, uma das áreas do AS é designada como a *área de backbone*, responsável por interconectar todas as áreas. Os roteadores na área de *backbone* são responsáveis por passar as informações coletadas de cada área para todas as outras. Desta forma, um roteador em uma área pode receber todos os LSPs gerados em outras. Para facilitar a comunicação, cada área recebe um identificador, sendo que o identificador de área do *backbone* vale zero. A Figura 4.76 mostra um sistema autônomo e suas áreas.

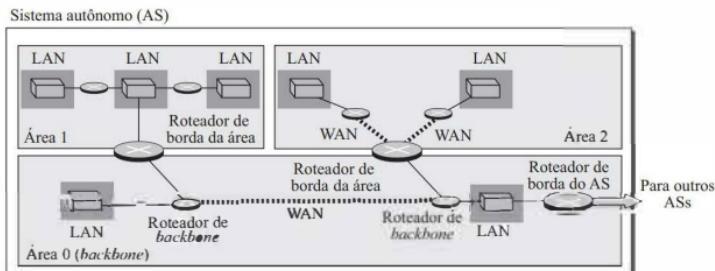



Figura 4.76 Áreas em um sistema autônomo.

Anúncio de estado de enlace

O OSPF é baseado no algoritmo de roteamento por estado de enlace, o qual requer que um roteador anuncie o estado de cada enlace a todos os vizinhos para a formação da LSDB. Quando discutimos o algoritmo de estado de enlace, usamos a teoria dos grafos e consideramos que cada roteador fosse um nó e que cada conexão entre dois roteadores fosse uma aresta. A situação é diferente no mundo real, onde é necessário anunciar a existência de entidades diferentes como nós, os tipos distintos de enlaces que ligam cada nó aos seus vizinhos e os diferentes tipos de custos associados a cada conexão. Isto significa que precisamos de diferentes tipos de anúncios, cada um capaz de anunciar situações distintas. Podemos ter cinco tipos de anúncios de estado de enlace: *enlace de roteador* (*router link*), *enlace de rede* (*network link*), *enlace resumo para rede* (*summary link to network*), *enlace resumo para roteador de borda do AS* (*summary link to AS border router*) e *enlace externo* (*external link*). A Figura 4.77 mostra esses cinco tipos de anúncio e seu uso.

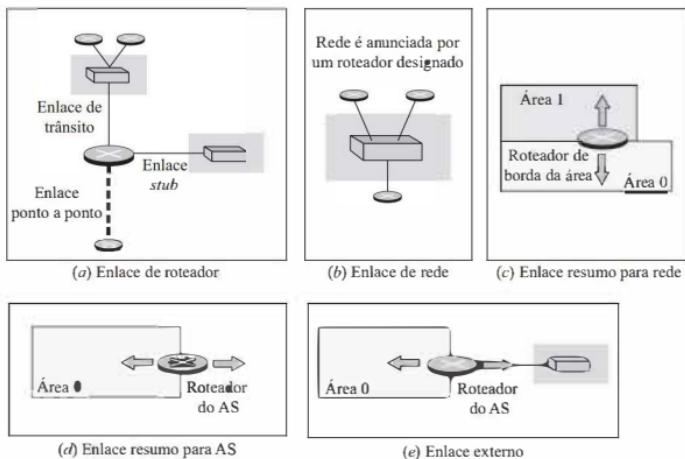



Figura 4.77 Cinco LSPs diferentes.

- **Enlace de roteador.** Além de fornecer o endereço do roteador de onde se origina o anúncio, esse tipo de anúncio pode especificar um ou mais tipos de enlaces que conectam o roteador fazendo o anúncio a outras entidades. Um *enlace de trânsito* anuncia uma conexão a uma rede de transição, uma rede que está conectada ao restante das redes por um ou mais roteadores. Esse tipo de anúncio deve especificar o endereço da rede de trânsito e o custo do enlace. Um *enlace stub* anuncia uma conexão a uma rede *stub*, uma rede que não liga outras redes. Novamente, o anúncio deve definir o endereço da rede e o custo. Um *enlace ponto a ponto* deve especificar o endereço do roteador localizado na extremidade da linha *ponto a ponto* e o custo para se chegar até lá.
- **Enlace de rede.** Um enlace de rede anuncia a rede como um nó. No entanto, como uma rede não pode ela mesma criar anúncios (ela é uma entidade passiva), um dos roteadores é escolhido como *roteador designado* e é ele quem envia os anúncios. Além do endereço do roteador designado, esse tipo de LSP anuncia o endereço IP de todos os roteadores (incluindo

o roteador designado como um roteador da rede, não apenas como uma entidade responsável por anúncios), mas nenhum custo é anunciado porque cada roteador anuncia o custo para a rede quando envia um anúncio de enlace de roteador, conforme discutido no item anterior.

- **Enlace resumo para rede.** Isto é feito por um roteador de borda da área, que divulga o resumo dos enlaces coletados pelo *backbone* para uma área, ou então o resumo dos enlaces coletados pela área para o *backbone*. Como discutimos anteriormente, essa troca de informações é necessária para interconectar as áreas.
- **Enlace resumo para AS.** Isto é feito por um roteador do AS que anuncia o resumo dos enlaces de outros AS para a área de *backbone* do AS corrente. Essas informações podem, depois, ser divulgadas para as áreas que não sejam de *backbone*, de modo que elas tomem conhecimento sobre as redes em outros AS. A necessidade desse tipo de troca de informações será melhor compreendida quando discutirmos um protocolo de roteamento inter-AS, o BGP.
- **Enlace externo.** Isto também é feito por um roteador do AS para anunciar a existência de uma única rede externa ao AS para a área de *backbone*, que então dissemina a informação para outras áreas.

Implementação do OSPF

O OSPF é implementado como um programa da camada de rede que usa os serviços do IP para a propagação de suas mensagens. Um datagrama IP que transporta uma mensagem OSPF preenche o campo de protocolo com o valor 89. Isto significa que, embora o OSPF seja um protocolo de roteamento que auxilia o IP a rotear seus datagramas dentro de um AS, as próprias mensagens OSPF são encapsuladas em datagramas. O OSPF possui duas versões: a versão 1 e a versão 2. A maioria das implementações atuais usa a versão 2.

Mensagens OSPF O OSPF é um protocolo muito complexo, que utiliza cinco tipos diferentes de mensagem. Na Figura 4.78, mostramos primeiramente o formato do cabeçalho OSPF em comum (ou seja, usado em todas as mensagens) e o cabeçalho genérico do estado de enlace (usado em algumas mensagens). Em seguida, fornecemos uma visão geral dos cinco tipos de mensagens utilizadas no OSPF. A mensagem de *olá* (*hello*: tipo 1) é usada por um roteador para se apresentar aos seus vizinhos e anunciar todos os vizinhos que ele já conhece. A mensagem de *descrição da base de dados* (*database description*: tipo 2) é normalmente enviada em resposta à mensagem de *olá* para permitir que um roteador recém-chegado adquira a LSDB completa. A mensagem de *solicitação de estado de enlace* (*link-state request*: tipo 3) é enviada por um roteador que precisa de informações sobre um estado de enlace específico. A mensagem de *atualização de estado de enlace* (*link-state update*: tipo 4) é a principal mensagem OSPF utilizada para a construção da LSDB. Essa mensagem, na verdade, apresenta cinco versões distintas (enlace de roteador, enlace de rede, enlace resumo para rede, enlace resumo para roteador de borda do AS e enlace externo), conforme discutido anteriormente. A mensagem de *confirmação de estado de enlace* (*link-state acknowledgment*: tipo 5) é usada para dar maior confiabilidade ao OSPF; todo roteador que recebe uma mensagem de atualização de estado de enlace precisa enviar uma confirmação de tal recebimento.

Autenticação Conforme mostra a Figura 4.78, o cabeçalho em comum do OSPF permite a autenticação do remetente da mensagem. Veremos no Capítulo 10 que isso impede que uma entidade maliciosa envie mensagens OSPF para um roteador com o objetivo de torná-lo parte de um sistema de roteamento ao qual ele, na verdade, não pertence.

Algoritmo do OSPF O OSPF implementa o algoritmo de roteamento por estado de enlace que discutimos na seção anterior. No entanto, algumas modificações e adendos precisam ser adicionados ao algoritmo:

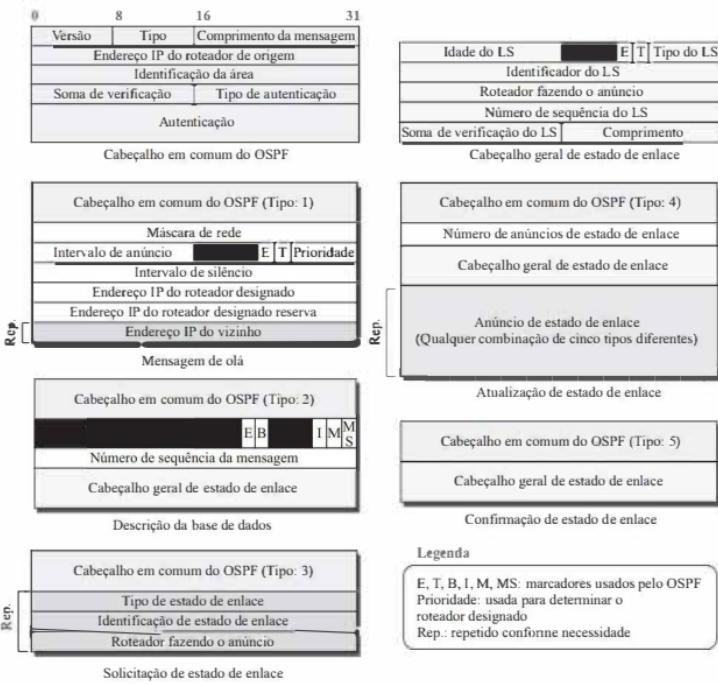



Figura 4.78 Formato das mensagens no OSPF.

- Depois de cada roteador ter criado sua árvore do menor custo, o algoritmo precisa usá-la para criar o algoritmo de roteamento correspondente.
- O algoritmo precisa ser expandido para lidar com o envio e recepção dos cinco tipos de mensagens mencionados anteriormente.

Desempenho

Antes de passarmos à próxima seção, discutiremos brevemente o desempenho do OSPF:

- **Update messages.** Mensagens de atualização. As mensagens de estado de enlace no OSPF têm um formato relativamente complexo. Elas também são enviadas para a área toda por meio de inundação. Se a área for grande, elas podem criar um tráfego elevado e usar muita banda.
- **Convergência das tabelas de roteamento.** Quando a inundação de LSPs é concluída, cada roteador pode criar sua própria árvore de caminho mais curto e sua própria tabela de roteamento; a convergência é bastante rápida. Entretanto, cada roteador precisa executar o algoritmo de Dijkstra, o que pode levar algum tempo.

- **Robustez.** O protocolo OSPF é mais robusto do que o RIP porque, depois de receber a LSDB completa, cada roteador torna-se independente de outros roteadores da área. Problemas de corrupção ou falha em um roteador não afetam outros roteadores de forma tão séria como ocorre no RIP.

Protocolo de Roteamento de Borda versão 4

O Protocolo de Roteamento de Borda versão 4 (BGP4 – Border Gateway Protocol version 4) é o único protocolo de roteamento interdomínios utilizado na Internet atual. O BGP4 baseia-se no algoritmo de vetor de caminhos descrito anteriormente, mas é adaptado para fornecer informações sobre a acessibilidade das redes na Internet.

Introdução

O BGP, e o BGP4 em particular, é um protocolo complexo. Nesta seção, apresentamos os fundamentos básicos do BGP e sua relação com protocolos de roteamento intradomínio (RIP ou OSPF). A Figura 4.79 mostra um exemplo de uma internet com quatro sistemas autônomos. AS2, AS3 e AS4 são sistemas autônomos *stub*; AS1 é um sistema autônomo de trânsito. No nosso exemplo, a troca de dados entre AS2, AS3 e AS4 deve passar por AS1.

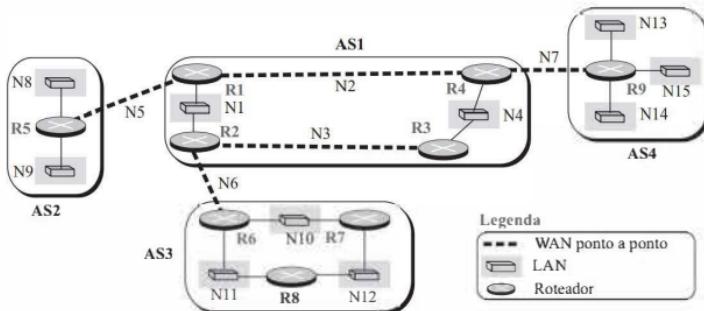



Figura 4.79 Uma internet-exemplo com quatro AS.

Cada sistema autônomo nesta figura usa um dos dois protocolos intradomínio mais comuns, RIP e OSPF. Cada roteador em cada AS sabe como chegar a uma rede que está em seu próprio AS, mas não sabe como chegar a uma rede em outro AS.

Para permitir que cada roteador roteie um pacote para qualquer rede na internet, precisamos, inicialmente, instalar uma variante do BGP4, denominada *BGP externo* (eBGP), em cada *roteador de borda* (o roteador localizado na fronteira de cada AS, conectado a um roteador em outro AS). Em seguida, instalamos a segunda variante da BGP, denominada *BGP interno* (iBGP), em todos os roteadores. Isto significa que os roteadores de borda executarão três protocolos de roteamento (intradomínio, eBGP e iBGP), enquanto os outros roteadores executarão dois protocolos (intradomínio e iBGP). Discutiremos o efeito de cada variante do BGP separadamente.

Operação do BGP externo (eBGP) Podemos dizer que o protocolo BGP é um tipo de protocolo ponto a ponto. Quando o *software* está instalado em dois roteadores, eles tentam criar uma conexão TCP usando a porta bem conhecida 179. Em outras palavras, um par de processos cliente e servidor

comunica-se continuamente para trocar mensagens. Os dois roteadores que executam os processos BGP são denominados *pares BGP* ou *interlocutores BGP*. Discutiremos os diferentes tipos de mensagens trocadas entre os dois pares, mas por ora estamos interessados apenas nas mensagens de atualização (discutidas mais adiante), que anunciam a acessibilidade das redes (ou seja, a possibilidade de se chegar a elas) em cada AS.

A variante eBGP de BGP permite que dois roteadores de borda em dois ASs distintos, porém conectados fisicamente, formem pares eBGP e troquem mensagens. Os roteadores que são elegíveis no exemplo da Figura 4.79 formam três pares: R1-R5, R2-R6 e R4-R9. A conexão entre esses pares é estabelecida sobre as três WANs físicas (N5, N6 e N7). No entanto, é necessário criar uma conexão lógica TCP sobre a conexão física para que a troca de informações seja possível. Cada conexão lógica, no jargão do BGP, é denominada *sessão*. Isto significa que precisamos de três sessões em nosso exemplo, conforme mostra a Figura 4.80.

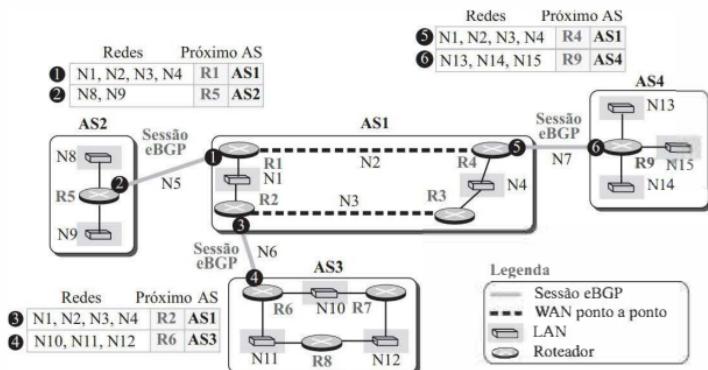



Figura 4.80 Operação do eBGP.

A figura também mostra as mensagens de atualização simplificadas enviadas pelos roteadores envolvidos nas sessões eBGP. O número dentro de um círculo indica o roteador responsável pelo envio em cada caso. Por exemplo, a mensagem número 1 é enviada pelo roteador R1 e diz ao roteador R5 que N1, N2, N3 e N4 podem ser alcançadas através do roteador R1 (R1 obtém essa informação da tabela de roteamento intradomínio correspondente). O roteador R5 pode agora adicionar essas informações ao final de sua tabela de roteamento. Quando R5 receber qualquer pacote destinado a essas quatro redes, pode usar sua tabela de roteamento e descobrir que o próximo roteador é R1.

O leitor deve ter notado que as mensagens trocadas durante as três sessões eBGP ajudam alguns roteadores a descobrir como rotear pacotes para algumas redes na internet, mas as informações sobre acessibilidade não estão completas. Há dois problemas que precisam ser resolvidos:

1. Alguns roteadores de borda não sabem como rotear um pacote destinado a AS não vizinhos. Por exemplo, R5 não sabe como rotear pacotes destinados às redes em AS3 e AS4. Os roteadores R6 e R9 estão na mesma situação que R5: R6 não tem informações sobre as redes em AS4, enquanto R9 não conhece as redes em AS3.
2. Exceto pelos roteadores de borda, nenhum roteador sabe como rotear um pacote destinado a redes em outros AS.

Para resolver esses dois problemas, precisamos permitir que todos os pares de roteadores (de borda ou não) executem a segunda variante do protocolo BGP, o iBGP.

Operação do BGP interno (iBGP) O protocolo iBGP é semelhante ao protocolo eBGP no sentido de que ele utiliza o serviço de TCP na porta conhecida 179; porém, cria uma sessão entre quaisquer pares possíveis de roteadores dentro de um sistema autônomo. No entanto, alguns pontos precisam ser esclarecidos. Primeiro, se um AS tem apenas um roteador, não pode haver uma sessão iBGP dentro dele. Por exemplo, não podemos criar uma sessão iBGP dentro do AS2 ou do AS4 na nossa internet. Segundo, se houver n roteadores em um sistema autônomo, deve haver $[n \times (n - 1)/2]$ sessões iBGP nesse sistema autônomo (uma malha totalmente conectada) para evitar laços no sistema. Em outras palavras, cada roteador precisa anunciar sua própria acessibilidade para seu par na sessão em vez de inundar a informação que ele recebe de outro par em outra sessão. A Figura 4.81 mostra a combinação das sessões eBGP e iBGP na nossa internet.

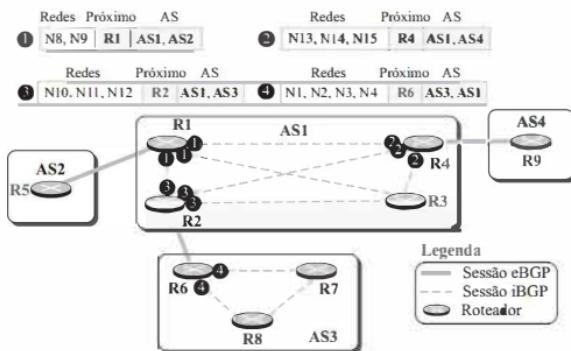



Figura 4.81 Combinação das sessões eBGP e iBGP na nossa internet.

Observe que não mostramos as redes físicas dentro dos AS porque uma sessão é criada usando uma rede de sobreposição (conexão TCP), que possivelmente se estende por mais de uma rede física, de acordo com a rota ditada pelo protocolo de roteamento intradomínio. Note também que nesse estágio apenas quatro mensagens são trocadas. A primeira mensagem (de número 1) é enviada por R1 anunciando que as redes N8 e N9 são acessíveis por meio do caminho AS1-AS2, mas que o próximo roteador é R1. Esta mensagem é enviada, usando sessões separadas, para R2, R3 e R4. Os roteadores R2, R4 e R6 fazem a mesma coisa, mas enviam mensagens diferentes para destinos diferentes. O ponto interessante é que, nesse estágio, R3, R7 e R8 criam sessões com os seus pares, mas, na verdade, não têm mensagens para enviar.

O processo de atualização não se encerra por aqui. Por exemplo, depois que R1 recebe a mensagem de atualização de R2, ele combina as informações de acessibilidade referentes a AS3 com as informações de acessibilidade que ele já possui referentes a AS1, e então envia uma nova mensagem de atualização para R5. Agora, R5 sabe como alcançar redes em AS1 e AS3. O processo continua quando R1 recebe a mensagem de atualização vinda de R4. Nesse processo, é importante nos certificarmos de que em algum momento não haverá mudanças nas atualizações anteriores e que todas as informações serão propagadas por todos os AS. Quando isso acontecer, cada roteador combina a informação recebida via eBGP e iBGP e cria o que podemos chamar de uma tabela de caminhos após a aplicação dos critérios para encontrar o melhor caminho, incluindo políticas de roteamento que discutiremos mais adiante. Mostramos, na Figura 4.82, as tabelas de caminhos para

Redes	Próximo	Caminho
N8, N9	R5	AS1, AS2
N10, N11, N12	R2	AS1, AS3
N13, N14, N15	R4	AS1, AS4

Tabela de caminho para R1

Redes	Próximo	Caminho
N8, N9	R1	AS1, AS2
N10, N11, N12	R6	AS1, AS3
N13, N14, N15	R1	AS1, AS4

Tabela de caminho para R2

Redes	Próximo	Caminho
N8, N9	R2	AS1, AS2
N10, N11, N12	R2	AS1, AS3
N13, N14, N15	R4	AS1, AS4

Tabela de caminho para R3

Redes	Próximo	Caminho
N8, N9	R1	AS1, AS2
N10, N11, N12	R1	AS1, AS3
N13, N14, N15	R9	AS1, AS4

Tabela de caminho para R4

Redes	Próximo	Caminho
N1, N2, N3, N4	R1	AS2, AS1
N10, N11, N12	R1	AS2, AS1, AS3
N13, N14, N15	R1	AS2, AS1, AS4

Tabela de caminho para R5

Redes	Próximo	Caminho
N1, N2, N3, N4	R2	AS3, AS1
N8, N9	R2	AS3, AS1, AS2
N13, N14, N15	R2	AS3, AS1, AS4

Tabela de caminho para R6

Redes	Próximo	Caminho
N1, N2, N3, N4	R6	AS3, AS1
N8, N9	R6	AS3, AS1, AS2
N13, N14, N15	R6	AS3, AS1, AS4

Tabela de caminho para R7

Redes	Próximo	Caminho
N1, N2, N3, N4	R6	AS3, AS1
N8, N9	R6	AS3, AS1, AS2
N13, N14, N15	R6	AS3, AS1, AS4

Tabela de caminho para R8

Redes	Próximo	Caminho
N1, N2, N3, N4	R4	AS4, AS1
N8, N9	R4	AS4, AS1, AS2
N10, N11, N12	R4	AS4, AS1, AS3

Tabela de caminho para R9

Figura 4.82 Tabelas de caminhos BGP finalizadas.

os roteadores da Figura 4.79. Por exemplo, o roteador R1 agora sabe que qualquer pacote destinado às redes N8 ou N9 deve passar por AS1 e AS2, e que o próximo roteador para o qual o pacote deve ser entregue é R5. Do mesmo modo, o roteador R4 sabe que qualquer pacote destinado às redes N10, N11 ou N12 deve passar por AS1 e AS3, e que o próximo roteador para o qual o pacote deve ser entregue é R1, e assim por diante.

Injeção de informações no roteamento intradomínio O papel de um protocolo de roteamento interdomínios tal como o BGP é ajudar os roteadores internos ao AS a expandir suas informações de roteamento. Ou seja, as tabelas de caminho coletadas e organizadas pelo BGP não são usadas, por si mesmas, no roteamento de pacotes; são injetadas nas tabelas de roteamento intradomínio (geradas via RIP ou OSPF) para permitir o roteamento de pacotes. Isto pode ser feito de várias maneiras, dependendo do tipo de AS.

No caso de um AS *stub*, o único roteador de borda da área adiciona uma entrada-padrão ao final de sua tabela de roteamento e designa o próximo roteador como o roteador interlocutor na outra extremidade da conexão eBGP. Na Figura 4.79, R5 em AS2 designa R1 como o roteador-padrão para todas as redes, com exceção de N8 e N9. A situação se repete para o roteador R9 em AS4, que designa R4 como roteador-padrão. Em AS3, R6 designa R2 como seu roteador-padrão, enquanto R7 e R8 designam R6 como roteador-padrão. Essas configurações estão em conformidade com as tabelas de caminhos que descrevemos na Figura 4.82 para esses roteadores. Em outras palavras, as tabelas de caminhos são injetadas nas tabelas de roteamento intradomínio por meio da inserção de apenas uma entrada-padrão.

No caso de um AS de trânsito, a situação é mais complicada. R1 em AS1 precisa injetar todo o conteúdo da tabela de caminhos para R1, mostrada na Figura 4.82, a sua tabela de roteamento intradomínio. A mesma situação se repete para R2, R3 e R4.

Uma questão a ser resolvida é o valor de custo. Sabemos que o RIP e o OSPF usam métricas diferentes. Uma solução muito comum é definir o custo das redes externas com o mesmo valor do custo para alcançar o primeiro AS no caminho. Por exemplo, o custo para que R5 alcance qualquer rede em outros AS equivale ao custo para alcançar N5. O custo para que R1 alcance as redes N10 a N12 equivale ao custo para alcançar N6, e assim por diante. O custo é obtido a partir das tabelas de roteamento intradomínio (RIP ou OSPF).

A Figura 4.83 mostra as tabelas de roteamento interdomínios. Para simplificar, consideramos que todos os AS estão usando RIP como o protocolo de roteamento intradomínio. As áreas sombreadas correspondem às extensões injetadas pelo protocolo BGP; os destinos-padrão são indicados como zero.

Tabela para R1

Des.	Próximo	Custo
N1	—	1
N4	R4	2
N8	R5	1
N9	R5	1
N10	R2	2
N11	R2	2
N12	R2	2
N13	R4	2
N14	R4	2
N15	R4	2

Tabela para R2

Des.	Próximo	Custo
N1	—	1
N4	R3	2
N8	R1	2
N9	R1	2
N10	R6	1
N11	R6	1
N12	R6	1
N13	R3	3
N14	R3	3
N15	R3	3

Tabela para R3

Des.	Próximo	Custo
N1	R2	2
N4	—	1
N8	R2	3
N9	R2	3
N10	R2	2
N11	R2	2
N12	R2	2
N13	R4	2
N14	R4	2
N15	R4	2

Tabela para R4

Des.	Próximo	Custo
N1	R1	2
N4	—	1
N8	R1	2
N9	R1	2
N10	R3	3
N11	R3	3
N12	R3	3
N13	R9	1
N14	R9	1
N15	R9	1

Tabela para R5

Des.	Próximo	Custo
N8	—	1
N9	—	1
0	R1	1

Tabela para R6

Des.	Próximo	Custo
N10	—	1
N11	—	1
N12	R7	2
0	R2	1

Tabela para R7

Des.	Próximo	Custo
N10	—	1
N11	R6	2
N12	—	1
0	R6	2

Tabela para R8

Des.	Próximo	Custo
N10	R6	2
N11	—	1
N12	—	1
0	R6	2

Tabela para R9

Des.	Próximo	Custo
N13	—	1
N14	—	1
N15	—	1
0	R4	1

Figura 4.83 Tabelas de roteamento após injeção de dados pelo BGP.

Agregação de endereços O leitor pode ter notado que as tabelas de roteamento intradomínio obtidas com a ajuda dos protocolos BGP4 podem tornar-se enormes no caso da Internet global, porque muitas redes de destino podem ser incluídas em uma tabela de roteamento. Felizmente, o BGP4 usa os prefixos como os identificadores de destino e permite sua agregação, conforme discutimos anteriormente neste capítulo. Por exemplo, os prefixos 14.18.20.0/26, 14.18.20.64/26, 14.18.20.128/26 e 14.18.20.192/26 podem ser combinados como 14.18.20.0/24 se todas as quatro sub-redes puderem ser alcançadas por meio de um mesmo caminho. Mesmo se um ou dois dos prefixos agregados precisarem de um caminho em separado, o princípio do maior prefixo discutido anteriormente nos permite fazer isso.

Atributos do caminho

Em ambos os protocolos de roteamento intradomínio discutidos aqui (RIP e OSPF), um destino é normalmente associado a duas informações: próximo salto e custo. A primeira informação diz o endereço do próximo roteador para o qual entregar o pacote, enquanto a segunda define o custo até o destino final. O roteamento interdomínios é mais complexo e, naturalmente, requer mais informações sobre como chegar ao destino final. No BGP, essas informações são denominadas **atributos de caminho**. O BGP permite que um destino seja associado a um máximo de sete atributos de caminho. Os atributos de caminho são divididos em duas grandes categorias: *bem conhecidos* e *opcionais*. Um atributo bem conhecido deve ser reconhecido por todos os roteadores, enquanto um atributo opcional não precisa sê-lo. Um atributo bem conhecido pode ser obrigatório, o que significa que ele deve estar presente em qualquer mensagem de atualização BGP, ou facultativo, ou seja, não é necessário. Um atributo opcional pode ser transitivo, o que significa que ele pode ser passado para o próximo AS, ou intransitivo, o que significa que isso não pode ser feito. Todos os atributos são inseridos após o prefixo de destino correspondente em uma mensagem de atualização (discutida mais adiante). O formato de um atributo é mostrado na Figura 4.84.

O primeiro *byte* em cada atributo define os quatro marcadores do atributo (conforme mostra a figura). O próximo *byte* define o tipo de atributo conforme especificado pela ICANN (apenas sete tipos foram especificados, conforme explicado a seguir). O comprimento do valor do atributo define o comprimento do campo de valor do atributo (não o comprimento de toda a seção de atributos). A seguir, apresentamos uma breve descrição de cada atributo.

O: *Bit* opcional (vale 1 se o atributo for opcional)
P: *Bit* parcial (vale 1 se o atributo for perdido em trânsito)
T: *Bit* transitivo (vale 1 se o atributo for transitivo)
E: *Bit* de extensão (vale 1 se o comprimento do atributo for dois *bytes*)

0					8	16	24 31
O	T	P	E	Todos 0s	Tipo de atributo	Comprimento do valor do atributo	
Valor do atributo (comprimento variável)							

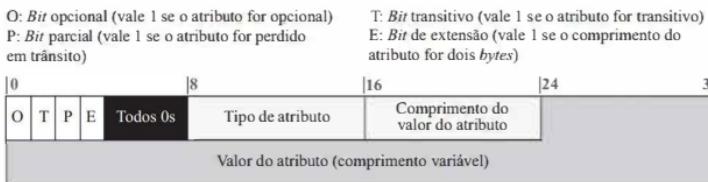



Figura 4.84 Formato dos atributos de caminho.

- **ORIGIN (tipo 1).** Este é um atributo bem conhecido e obrigatório, que define a origem da informação de roteamento. Pode assumir um dos três valores: 1, 2 ou 3. O valor 1 significa que a informação sobre o caminho foi obtida de um protocolo intradomínio (RIP ou OSPF). O valor 2 significa que a informação vem do BGP. O valor 3 significa que ela vem de uma fonte desconhecida.
- **AS-PATH (tipo 2).** Este é um atributo bem conhecido e obrigatório, que define a lista de sistemas autônomos através dos quais o destino pode ser alcançado. Chegamos a usar esse atributo em nossos exemplos. O atributo AS-PATH, ajuda a evitar laços, conforme discutimos no roteamento por vetor de caminhos na seção anterior. Sempre que uma mensagem de atualização que chega a um roteador contém o AS corrente em sua lista, o roteador descarta tal caminho. O atributo AS-PATH também pode ser utilizado na seleção de rotas.
- **NEXT-HOP (tipo 3).** Este é um atributo bem conhecido e obrigatório, que define o próximo roteador para o qual o pacote de dados deve ser enviado (próximo salto). Também chegamos a utilizá-lo em nossos exemplos. Como vimos, esse atributo ajuda a injetar informações sobre caminhos coletadas por meio das operações do eBGP e do iBGP nos protocolos de roteamento intradomínio, como o RIP e o OSPF.
- **MULT-EXIT-DISC (tipo 4).** O discriminador de múltiplas saídas é um atributo não transitivo opcional que distingue entre múltiplos caminhos de saída para um destino. O valor desse atributo é normalmente definido pela métrica no protocolo intradomínio correspondente (um valor de atributo na forma de um inteiro sem sinal de quatro *bytes*). Por exemplo, se um roteador tem múltiplos caminhos para certo destino com diferentes valores relacionados a esses atributos, aquele com o menor valor é selecionado. Perceba que esse atributo é não transitivo, ou seja, não é propagado de um AS para outro.
- **LOCAL-PREF (tipo 5).** O atributo de preferência local é um atributo bem conhecido facultativo. Costuma ser definido pelo administrador, com base na política da organização. As rotas que o administrador preferir recebem um valor mais alto de preferência local (um valor de atributo na forma de um inteiro sem sinal de quatro *bytes*). Por exemplo, em uma internet com cinco AS, o administrador do AS1 pode definir o valor da preferência local como 400 para o caminho AS1-AS2-AS5, como 300 para AS1-AS3-AS5 e como 50 para AS1-AS4-AS5. Isto significa que o administrador prefere o primeiro caminho em detrimento do segundo e prefere o segundo em vez do terceiro. Pode ser que o AS2 seja considerado o mais seguro dos sistemas autônomos e o AS4, o menos seguro deles pela administração do AS1. A última rota deve ser escolhida se as outras duas não estiverem disponíveis.
- **ATOMIC-AGGREGATE (tipo 6).** Este é um atributo bem conhecido facultativo, que especifica que o prefixo de destino não é agregado, ou seja, define uma única rede de destino. Esse atributo não tem um campo de valor, ou seja, o valor do campo de comprimento é igual a zero.

- **AGGREGATOR (tipo 7).** Este é um atributo opcional transitivo que enfatiza que o prefixo de destino é um agregado. O valor do atributo fornece o número do último AS que realizou a agregação seguido pelo endereço IP do roteador que o fez.

Seleção de rota

Nesta seção, ainda não discutimos sobre como uma rota é selecionada por um roteador BGP, principalmente porque nosso exemplo simples tem apenas uma rota para um destino. Caso múltiplas rotas para um destino sejam recebidas, o BGP precisa selecionar uma delas. O processo de seleção de rota no BGP não é tão fácil quanto nos protocolos de roteamento intradomínio baseados na árvore de caminho mais curto. Uma rota no BGP tem alguns atributos associados a ela, e pode ser o resultado de uma sessão eBGP ou de uma sessão iBGP. A Figura 4.85 mostra o diagrama de fluxo tal como utilizado por implementações comuns.

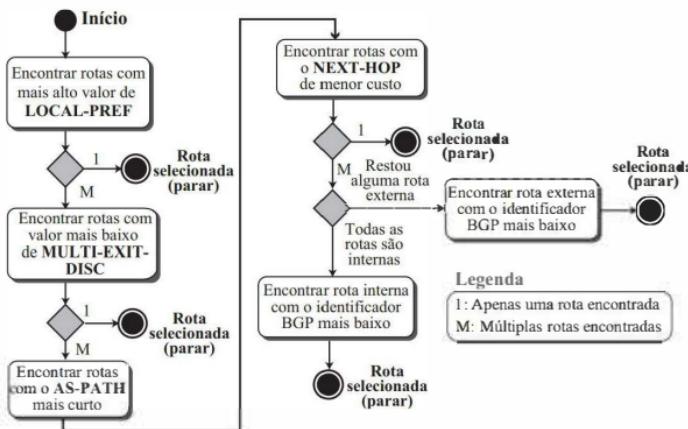



Figura 4.85 Diagrama de fluxo para a seleção de rota.

O roteador extrai as rotas que satisfazem os critérios de cada passo. Se apenas uma rota for extraída, ela é selecionada e o processo se encerra; caso contrário, o processo continua com o próximo passo. Perceba que a primeira escolha está relacionada ao atributo LOCAL-PREF, que reflete a política imposta pela administração sobre a rota.

Mensagens

O BGP usa quatro tipos de mensagens para a comunicação entre os interlocutores BGP em todos os AS e dentro de cada AS: *abertura* (*open*), *atualização* (*update*), *manutenção da sessão* (*keepalive*) e *notificação* (*notification*), conforme mostrado na Figura 4.86. Todos os pacotes BGP compartilham o mesmo cabeçalho em comum.

- **Mensagem de abertura.** Para criar uma relação de vizinhança, um roteador executando BGP abre uma conexão TCP com um vizinho e envia uma *mensagem de abertura* (*open*).

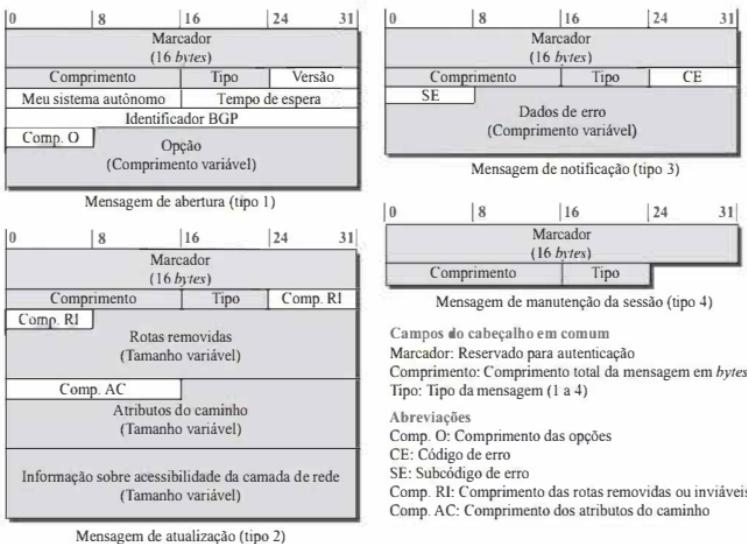



Figura 4.86 Mensagens BGP.

- **Mensagem de atualização.** A *mensagem de atualização* (*update*) é o coração do protocolo BGP. Ela é usada por um roteador para remover destinos que tenham sido anunciados anteriormente, para anunciar uma rota para um novo destino, ou ambos. Perceba que o BGP pode remover vários destinos que foram anunciados anteriormente, mas só pode anunciar um novo destino (ou vários destinos com os mesmos atributos de caminho) em cada mensagem de atualização.
- **Mensagem de manutenção da sessão.** Os pares BGP ativos trocam mensagens de manutenção da sessão (*keepalive*), antes que seu tempo de espera chegue a expirar, para informar uns aos outros que eles continuam ativos.
- **Notificação.** Uma mensagem de notificação (*notification*) é enviada por um roteador sempre que uma condição de erro é detectada ou que um roteador queira fechar a sessão.

Desempenho

O desempenho BGP pode ser comparado com o do RIP. Os interlocutores BGP trocam um grande número de mensagens para criar tabelas de roteamento, mas BGP é livre de laços e do problema da contagem até o infinito. A mesma fraqueza que mencionamos para o RIP relativa à propagação de falhas e corrupções também existe no BGP.

4.4 ROTEAMENTO *MULTICAST*

A comunicação na Internet atualmente não consiste apenas em *unicast*; a comunicação via *multicast* está crescendo rapidamente. Nesta seção, primeiramente discutimos as ideias gerais por trás

do *unicast*, do *multicast* e do *broadcast*. Em seguida, abordamos algumas questões básicas que aparecem no roteamento *multicast*. Finalmente, discutimos os protocolos de roteamento *multicast* na Internet.

4.4.1 Introdução

Nas seções anteriores, aprendemos que o encaminhamento de datagramas por um roteador é normalmente realizado com base no prefixo do endereço de destino do datagrama, que determina a rede à qual a estação de destino está conectada. Após compreender o princípio de encaminhamento anterior, podemos agora definir as comunicações *unicast*, *multicast* e *broadcast*. Esclareceremos esses termos e como eles se relacionam com a Internet.

Comunicação *unicast*

Na comunicação *unicast*, existe apenas uma rede de origem e uma de destino. A relação entre a origem e o destino da rede é um para um. Cada roteador no caminho do datagrama tenta encaminhar o pacote para apenas uma de suas interfaces. A Figura 4.87 mostra uma pequena internet na qual um pacote *unicast* precisa ser entregue de um computador de origem para um computador de destino conectado a N6. O roteador R1 é responsável por encaminhar o pacote somente através da interface 3; o roteador R4 é responsável por encaminhar o pacote somente através da interface 2. Quando o pacote chega a N6, a entrega à estação de destino é de responsabilidade da rede; o pacote pode ser transmitido a todas as estações, ou o *switch* Ethernet pode entregá-lo apenas à estação de destino.

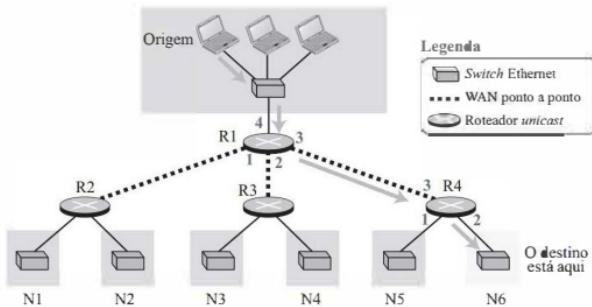



Figura 4.87 Comunicação *unicast*.

Comunicação *multicast*

Na comunicação *multicast*, existe uma origem e um grupo de destinos. A relação é um para muitos. Nesse tipo de comunicação, o endereço de origem é um endereço *unicast*, mas o de destino é um endereço de grupo, ou seja, um grupo de uma ou mais redes de destino nas quais há pelo menos um membro do grupo que esteja interessado em receber o datagrama *multicast*. O endereço do grupo define seus membros. A Figura 4.88 mostra a mesma internet de tamanho reduzido apresentada na Figura 4.87, mas cujos roteadores foram trocados por roteadores *multicast* (ou os roteadores anteriores foram configurados para suportar ambos os tipos de comunicação).

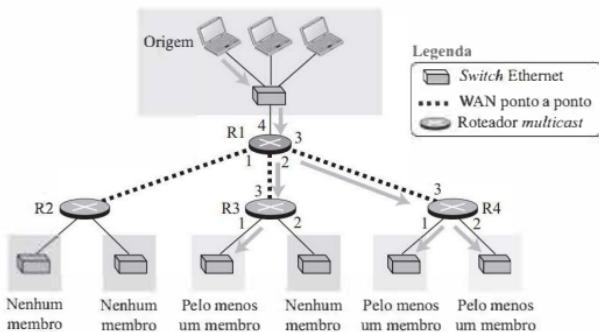



Figura 4.88 Comunicação *multicast*.

Quando se usa *multicast*, um roteador *multicast* pode precisar enviar cópias do mesmo datagrama por mais de uma interface. Na Figura 4.88, o roteador R1 deve enviar o datagrama pelas interfaces 2 e 3. De forma similar, o roteador R4 precisa enviar os datagramas por ambas as suas interfaces. O roteador R3, no entanto, sabe que não há membros que pertencem a esse grupo na área alcançável pela interface 2; assim, ele envia o datagrama apenas pela interface 1.

Comunicação *multicast versus* múltiplas comunicações *unicast*

Precisamos deixar clara a diferença entre uma comunicação *multicast* e múltiplas comunicações *unicast*. A Figura 4.89 ilustra ambos os conceitos.

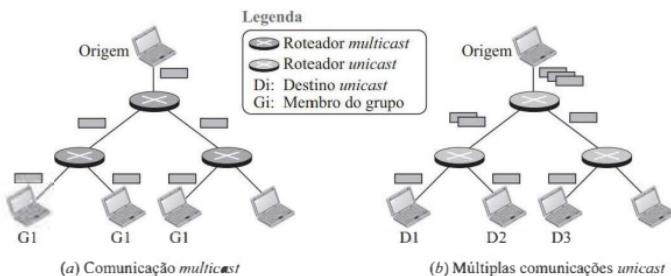



Figura 4.89 Comunicação *multicast versus* múltiplas comunicações *unicast*.

A comunicação *multicast* começa com um único pacote vindo da origem que é duplicado pelos roteadores. O endereço de destino em cada pacote é o mesmo para todas as duplicatas. Perceba que apenas uma cópia do pacote trafega entre dois roteadores.

Quando se usam múltiplas comunicações *unicast*, vários pacotes saem da origem. Se houver três destinos, por exemplo, a origem envia três pacotes, cada um com um endereço de destino *unicast* distinto. Perceba que pode haver várias cópias trafegando entre dois roteadores. Por exemplo,

quando alguém envia uma mensagem de e-mail para um grupo de pessoas, isto equivale a múltiplos envios *unicast*. O aplicativo de e-mail cria réplicas da mensagem, cada uma com um endereço de destino diferente, e as envia uma a uma.

Emulação do *multicast* via *unicast*

Podemos nos perguntar por que existe um mecanismo separado para realizar *multicast*, quando é possível emulá-lo via *unicast*. Existem pelo menos duas razões:

1. Usar *multicast* é mais eficiente que usar múltiplos *unicasts*. Na Figura 4.89, podemos ver como o *multicast* requer menos largura de banda que múltiplos *unicasts*. Em uma comunicação com múltiplos *unicasts*, alguns dos enlaces devem tratar várias cópias da mensagem.
2. Em uma comunicação com múltiplos *unicasts*, os pacotes são criados pelo remetente com um certo atraso entre pacotes. Se houver mil destinos, o atraso entre o primeiro e o último pacote pode se tornar inaceitável. No *multicast*, não há qualquer atraso porque apenas um pacote é criado pelo remetente.

Aplicações do *multicast*

O *multicast* tem muitas aplicações atualmente, como o acesso a bancos de dados distribuídos, disseminação de informações, teleconferência e ensino à distância.

- **Acesso a bancos de dados distribuídos.** A maioria dos atuais bancos de dados de grande porte é distribuída. Isto é, as informações são armazenadas em mais de um local, geralmente no momento em que são produzidas. O usuário que precisa acessar o banco de dados não sabe a localização das informações. Uma solicitação do usuário é enviada via *multicast* para todos os locais nos quais o banco de dados está localizado, e então o local no qual a informação se encontra responde à solicitação.
- **Disseminação de informações.** Muitas empresas comumente precisam enviar informações para os seus clientes. Se a natureza da informação é igual para cada cliente, isto pode ser feito via *multicast*. Assim, a empresa pode enviar uma única mensagem capaz de atingir muitos clientes. Por exemplo, uma atualização de *software* pode ser enviada a todos os compradores de um pacote de *software* específico. De maneira similar, uma notícia pode ser facilmente disseminada via *multicast*.
- **Teleconferência.** Teleconferências envolvem o uso de *multicast*. As pessoas participando de uma teleconferência precisam receber a mesma informação ao mesmo tempo. Grupos *multicast* temporários ou permanentes podem ser formados para essa finalidade.
- **Ensino à distância.** Uma área crescente de utilização de *multicast* é o ensino à distância. As aulas ministradas por um professor podem ser recebidas por um grupo específico de alunos. Isto é especialmente conveniente àqueles alunos que têm dificuldade para frequentar presencialmente as aulas no *campus*.

Comunicação *broadcast*

A comunicação via *broadcast* é um tipo de comunicação um para todos: um *host* envia um pacote para todos os *hosts* em uma internet. A comunicação *broadcast*, nesse sentido, não é fornecida no nível da Internet pela razão óbvia de que isso poderia criar um volume de tráfego enorme e utilizar uma gigantesca quantidade de largura de banda. O *broadcast* parcial, no entanto, é possível na Internet. Por exemplo, algumas aplicações *peer-to-peer* podem fazer uso de *broadcast* para acessar todos os *peers* da rede. O *broadcast* controlado também pode ser realizado em um domínio (área ou sistema autônomo), geralmente como uma etapa para se criar uma comunicação *multicast*. Discutiremos esse tipo de *broadcast* controlado quando abordarmos protocolos de *multicast*.

4.4.2 Fundamentos do *multicast*

Antes de discutirmos protocolos de roteamento *multicast* na Internet, precisamos discutir algumas noções básicas de *multicast*: endereçamento *multicast*, coleta de informações sobre grupos *multicast* e árvores de *multicast* ideais.

Endereços *multicast*

Quando enviamos um pacote *unicast* para um destino, o endereço de origem do pacote define o remetente e o endereço de destino do pacote define o seu destinatário. Na comunicação *multicast*, há apenas um remetente, mas existem vários destinatários, às vezes milhares ou milhões espalhados pelo mundo todo. Deve ficar claro que não é possível incluir os endereços de todos os destinatários no pacote. O endereço de destino de um pacote, tal como descrito no Protocolo Internet (IP) deve ser apenas um. Por isso, precisamos de endereços *multicast*. Um endereço *multicast* define um grupo de destinatários, não apenas um. Em outras palavras, um endereço *multicast* é um identificador para um grupo. Se um novo grupo é formado por alguns membros ativos, uma autoridade pode atribuir um endereço *multicast* que não esteja em uso para esse grupo de modo a defini-lo univocamente. Isto significa que o endereço de origem de um pacote em uma comunicação *multicast* pode ser um endereço *unicast* que define exclusivamente o remetente, mas o endereço de destino será o endereço *multicast* que define um grupo. Assim, uma estação que seja membro de *n* grupos, na verdade, terá (*n* + 1) endereços: um endereço *unicast* que é usado como endereço de origem ou de destino em comunicações *unicast* e *n* endereços *multicast* usados apenas como endereço de destino para receber mensagens enviadas para um grupo. A Figura 4.90 mostra esse conceito.

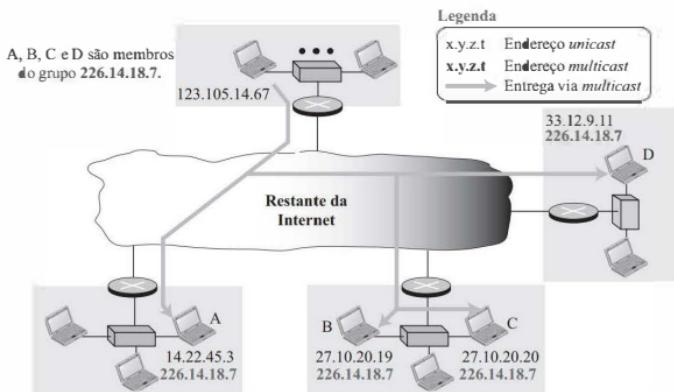



Figura 4.90 Necessidade de endereços *multicast*.

Endereços *multicast* no IPv4

Um roteador ou uma estação de destino deve ser capaz de distinguir entre datagramas *unicast* e *multicast*. O IPv4 e o IPv6 atribuem um bloco de endereços para essa finalidade. Nesta seção, discutimos apenas endereços *multicast* IPv4; os endereços *multicast* IPv6 serão discutidos mais adiante neste capítulo. Os endereços *multicast* no IPv4 pertencem a um grande bloco de endereços

que foram projetados especialmente com esse propósito. No endereçamento com classes, a classe D inteira era composta por tais endereços; o endereçamento sem classes usava o mesmo bloco, que passou a ser identificado como o bloco 224.0.0.0/4 (de 224.0.0.0 a 239.255.255.255). A Figura 4.91 mostra o bloco em binário. Quatro *bits* definem o bloco; o restante dos *bits* é utilizado como o identificador do grupo.

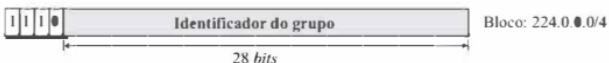



Figura 4.91 Um endereço *multicast* em binário.

O número de endereços no bloco de *multicast* é enorme (2^{28}). Definitivamente, não deve haver um número tão grande de grupos individuais. No entanto, o bloco é dividido em vários sub-blocos, e cada sub-bloco é usado em uma aplicação *multicast* em particular. A seguir, apresentamos alguns dos sub-blocos comuns:

- **Bloco de controle da rede local.** O sub-bloco 224.0.0.0/24 é reservado a protocolos de roteamento *multicast* utilizados dentro de uma rede, o que significa que pacotes cujo endereço de destino se encontre nessa faixa não podem ser encaminhados por um roteador. Nesse sub-bloco, o endereço 224.0.0.0 é reservado, o endereço 224.0.0.1 é usado para enviar datagramas para todas as estações e roteadores dentro de uma rede, e o endereço 224.0.0.2 é utilizado para enviar datagramas para todos os roteadores dentro de uma rede. Os endereços restantes são atribuídos a alguns protocolos de *multicast* para comunicação, como discutiremos mais adiante.
- **Bloco de controle inter-redes.** O sub-bloco 224.0.1.0/24 é reservado a protocolos de roteamento *multicast* usados em toda a Internet, o que significa que pacotes cujo endereço de destino se encontre nessa faixa podem ser repassados por um roteador.
- **Bloco multicast de origem específica (SSM – Source-Specific Multicast).** O bloco 232.0.0.0/8 é utilizado para roteamento *multicast* de origem específica. Discutiremos o roteamento SSM quando descrevermos o protocolo IGMP no final do capítulo.
- **Bloco GLOP.** O bloco 233.0.0.0/8 é denominado bloco GLOP (não é uma sigla nem uma abreviatura). Esse bloco define uma faixa de endereços que podem ser usados dentro de um sistema autônomo (AS). Como vimos anteriormente, a cada sistema autônomo é atribuído um número de 16 *bits*. Pode-se utilizar o número do AS como os dois octetos do meio no bloco para criar uma faixa de 256 endereços *multicast* (233.x.y0 até 233.x.y.255), onde x.y é o número do AS.
- **Bloco de alcance administrativo.** O bloco 239.0.0.0/8 é chamado de Bloco de Alcance Administrativo. Os endereços desse bloco são utilizados em uma área específica da Internet. Os pacotes cujo endereço de destino pertence a esse intervalo supostamente não devem deixar a área. Em outras palavras, um endereço nesse bloco é restrito a uma organização.

Selecionando o endereço *multicast*

Selecionar um endereço *multicast* a ser atribuído a um grupo não é uma tarefa fácil. A seleção do endereço depende do tipo de aplicação. Discutiremos alguns casos.

Grupo limitado O administrador da rede pode usar o número do AS (x.y)256 e escolher um endereço entre 239.x.y.0 e 239.x.y.255 (Bloco de Alcance Administrativo) que não esteja em

uso por qualquer outro grupo, como o endereço *multicast* para aquele grupo em particular. Por exemplo, considere que professores universitários precisam criar endereços de grupo para a comunicação com seus alunos. Se o número do AS ao qual a universidade pertence é 23452, que pode ser escrito como (91.156) 256, isso dá à universidade uma faixa de 256 endereços: 233.91.156.0 a 233.91.156.255. A administração da faculdade pode conceder a cada professor um dos endereços nesse intervalo. Esse endereço pode, então, tornar-se o endereço do grupo que o professor usará para enviar mensagens via *multicast* para os alunos. No entanto, os pacotes não são capazes de deixar o território do AS da universidade.

Grupo maior Se o grupo se estende além do território de um AS, a solução anterior não funciona. O grupo precisa escolher um endereço do bloco SSM (232.0.0.8). Não é necessário obter permissão para usar um endereço desse bloco, pois os pacotes no *multicast* de origem específica são roteados com base no grupo e no endereço de origem; eles são únicos.

Coletando informações sobre os grupos

A criação das tabelas de roteamento tanto no roteamento *unicast* como no *multicast* envolve duas etapas:

1. Um roteador precisa saber a quais destinos ele está conectado.
2. Cada roteador precisa propagar as informações obtidas na primeira etapa para todos os outros roteadores, de modo que cada roteador saiba a qual destino cada outro roteador está conectado.

No roteamento *unicast*, a coleta de informações na primeira etapa é automática; cada roteador sabe a qual rede ele está conectado, e o prefixo da rede (no CIDR) é aquilo de que um roteador necessita. Os protocolos de roteamento que descrevemos nas seções anteriores (vetor de distâncias e estado de enlace) são responsáveis pela propagação dessas informações, coletadas automaticamente, para todos os outros roteadores na internet.

No roteamento *multicast*, a coleta de informações na primeira etapa não é automática por duas razões. Primeiro, um roteador não sabe qual estação na rede a ele conectada é membro de um grupo em particular; a participação no grupo não tem qualquer relação com o prefixo associado à rede. Mostramos que, se uma estação é membro de um grupo, ela tem um endereço *multicast* em separado relacionado a tal grupo. Segundo, a participação em um grupo não é um atributo fixo de uma estação; ela pode entrar ou sair de alguns grupos em um curto período de tempo. Por isso, um roteador precisa de ajuda para descobrir quais grupos estão ativos em cada uma de suas interfaces. Após a coleta dessas informações, o roteador pode propagá-las para qualquer outro roteador usando um protocolo de roteamento *multicast*, conforme discutiremos mais adiante. A Figura 4.92 mostra a diferença entre os anúncios *unicast* e *multicast* na primeira etapa. Para o *unicast*, o roteador não precisa de ajuda; já para o *multicast*, precisa da ajuda de um outro protocolo, conforme discutiremos a seguir.

No caso do *unicast*, o roteador R sabe que as estações com prefixos N1, N2 e N3 estão conectadas às suas interfaces; ele propaga essas informações para o restante da internet. No caso do *multicast*, o roteador R precisa saber que existem máquinas com pelo menos um membro ativo nos grupos G1, G2 e G3 nas redes conectadas às suas interfaces. Em outras palavras, para o roteamento *unicast*, precisamos apenas que o protocolo de roteamento interno de cada domínio propague as informações sobre um enlace do roteador; já no *multicast*, precisamos de dois protocolos: um para coletar essas informações e o segundo para propagá-las. Antes de detalharmos os protocolos envolvidos na segunda etapa, precisamos discutir o protocolo envolvido na primeira.

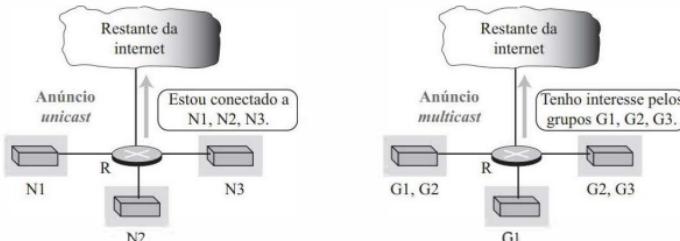



Figura 4.92 Anúncios *unicast versus multicast*

Protocolo de Gerenciamento de Grupos Internet

O protocolo utilizado atualmente para a coleta de informações sobre a associação de uma estação a um grupo é o **Protocolo de Gerenciamento de Grupos Internet** (IGMP – Internet Group Management Protocol). O IGMP é um protocolo da camada de rede; é um dos protocolos auxiliares que, assim como o ICMP, é considerado parte do IP. As mensagens IGMP, tais como as mensagens ICMP, são encapsuladas em datagramas IP. Existem apenas dois tipos de mensagens no IGMP versão 3: mensagens de consulta e de relatório, conforme mostra a Figura 4.93.

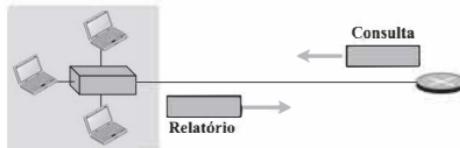



Figura 4.93 Operação do IGMP.

- **Mensagem de consulta.** Uma mensagem de consulta é periodicamente enviada por um roteador para todos os *hosts* ligados a ele para pedir-lhes que relatem seus interesses sobre a participação em grupos. No IGMPv3, uma mensagem de consulta pode assumir uma de três formas:
  - **a.** Uma mensagem de consulta *geral* refere-se à participação em qualquer grupo. Ela é encapsulada em um datagrama com o endereço de destino 224.0.0.1 (todas as estações e roteadores). Perceba que todos os roteadores conectados à mesma rede recebem essa mensagem para informá-los que tal mensagem já foi enviada e que eles devem abster-se de reenviá-la.
  - **b.** Um mensagem de consulta de *grupo específico* é enviada por um roteador para perguntar sobre a participação relativa a um grupo específico. Ela é enviada quando um roteador não recebe uma resposta sobre um grupo específico e quer ter certeza de que não há qualquer membro ativo desse grupo na rede. O identificador do grupo (endereço *multicast*) é especificado na mensagem, que é encapsulada em um datagrama com o endereço de destino preenchido com o endereço *multicast* correspondente. Apesar de todas as estações receberem essa mensagem, ela é descartada pelas estações que não estão interessadas no grupo consultado.

c. Uma mensagem de consulta de *origem e grupo específicos* é enviada por um roteador para interrogar sobre a participação em um grupo específico quando a mensagem vem de uma origem ou origens específicas. Mais uma vez, ela é enviada quando o roteador não recebe informações sobre um grupo específico com relação a uma determinada estação ou estações. A mensagem é encapsulada em um datagrama com o endereço de destino definido como o endereço *multicast* correspondente. Apesar de todas as estações receberem essa mensagem, aquelas que não estão interessadas no grupo consultado a descartam.

- **Mensagem de relatório.** Uma mensagem de relatório é enviada por uma estação em resposta a uma mensagem de consulta. Ela contém uma lista de registros, e cada um fornece o identificador do grupo correspondente (endereço *multicast*) e os endereços de todas as origens das quais a estação está interessada em receber mensagens (inclusão). O registro também pode especificar os endereços de origem dos quais a estação não deseja receber mensagens destinadas ao grupo (exclusão). A mensagem é encapsulada em um datagrama com o endereço *multicast* 224.0.0.22 (endereço *multicast* atribuído ao IGMPv3).

No IGMPv3, se uma estação deseja participar de um grupo, ela espera até receber uma mensagem de consulta e então envia uma mensagem de relatório. Se uma estação deseja sair de um grupo, ela não responde a uma mensagem de consulta. Se nenhuma outra estação responder à mensagem correspondente, o grupo é removido da base de dados do roteador.

Encaminhamento *multicast*

Outra questão importante no *multicast* diz respeito à decisão que um roteador precisa tomar sobre encaminhar um pacote *multicast*. O encaminhamento nas comunicações *unicast* e *multicast* difere em dois aspectos:

1. Na comunicação *unicast*, o endereço de destino do pacote define um único destino. O pacote deve ser enviado somente por uma das interfaces do roteador, correspondente ao ramo da árvore de menor custo que permite chegar ao destino com o mínimo custo envolvido. Na comunicação *multicast*, o destino do pacote define um grupo, mas pode haver mais de um membro daquele grupo na internet. Para alcançar todos os destinos, o roteador pode ser obrigado a enviar o pacote por mais de uma interface. A Figura 4.94 ilustra esse conceito. No *unicast*, a rede de destino N1 não pode estar em mais de uma parte da internet; já no *multicast*, pode haver membros do grupo G1 em mais de uma parte da internet.

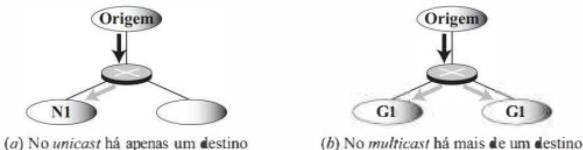



Figura 4.94 Destinos no *unicast* e no *multicast*.

2. As decisões sobre encaminhamento na comunicação *unicast* dependem apenas do endereço de destino do pacote. As decisões de encaminhamento na comunicação *multicast* dependem tanto do endereço de destino como da origem do pacote. Em outras palavras, no *unicast*, o encaminhamento é feito com base em aonde o pacote deve ir; já no *multicast*, o encaminhamento se baseia em aonde o pacote deve ir e de onde o pacote veio. A Figura

4.95 ilustra esse conceito. Na parte (*a*) da figura, a origem está em uma região da internet onde não há membros do grupo. Na parte (*b*), a origem está em uma região onde existe um membro do grupo. Na parte (*a*), o roteador precisa enviar o pacote por duas interfaces; na parte (*b*), o roteador deve enviar o pacote por uma só interface para evitar o envio de uma segunda cópia do pacote pela interface de onde ele chegou. Ou seja, na parte (*b*) da figura, o membro (ou membros) do grupo G1 já recebeu uma cópia do pacote antes de tal pacote chegar ao roteador; enviar o pacote naquela direção é desnecessário, além de gerar mais tráfego. Isso mostra que o encaminhamento na comunicação *multicast* depende tanto dos endereços de origem como de destino.

(*a*) Pacote enviado por duas interfaces (*b*) Pacote enviado por uma interface

Figura 4.95 Encaminhamento *multicast* depende do destino e da origem.

Duas abordagens na comunicação *multicast*

No roteamento *multicast*, assim como no roteamento *unicast*, precisamos criar árvores de roteamento para otimizar a rota dos pacotes da origem até seu destino. No entanto, conforme discutimos anteriormente, a decisão relativa ao roteamento *multicast* em cada roteador depende não apenas do destino do pacote, mas também de sua origem. O envolvimento da origem no processo de roteamento torna o roteamento *multicast* muito mais difícil que o *unicast*. Por esse motivo, foram desenvolvidas duas abordagens diferentes de roteamento *multicast*: roteamento usando árvores baseadas na origem e roteamento usando árvores compartilhadas pelo grupo.

Abordagem de árvores baseadas na origem

Na abordagem de *multicast* usando árvores baseadas na origem, cada roteador precisa criar uma árvore separada para cada combinação origem-grupo. Em outras palavras, se existem m grupos e n origens na internet, um roteador precisa criar $(m \times n)$ árvores de roteamento. Em cada árvore, a origem correspondente é a raiz, os membros do grupo são as folhas, e o próprio roteador está em algum lugar na árvore. Podemos comparar essa situação com o roteamento *unicast*, em que um roteador precisa de uma só árvore na qual ele mesmo é a raiz e todas as redes na internet são as folhas. Embora possa parecer que cada roteador precise criar e armazenar uma quantidade enorme de informações sobre todas essas árvores, existem dois protocolos que utilizam essa abordagem na Internet atualmente, algo que discutiremos mais adiante. Os protocolos adotam algumas estratégias para aliviar essa situação.

Abordagem de árvores compartilhadas pelo grupo

Na abordagem de árvores compartilhadas pelo grupo, designamos um roteador para atuar como a falsa origem para cada grupo. O roteador designado, denominado roteador *central* (*core*) ou roteador de ponto de encontro (*rendezvous-point*), atua como um representante do grupo. Qualquer máquina que tenha um pacote para enviar a um membro desse grupo envia o pacote para o roteador central (comunicação *unicast*), que fica responsável por realizar o *multicast*. O roteador central cria uma única árvore de roteamento tendo a si mesmo como raiz e quaisquer roteadores com membros ativos do grupo como folhas. Nessa abordagem, existem m roteadores centrais (um para cada grupo) e

cada roteador central tem uma árvore de roteamento, levando a um total de *m* árvores. Isto significa que o número de árvores de roteamento é reduzido de ($m \times n$) na abordagem de árvores baseadas na origem para *m* nessa abordagem. O leitor deve ter notado que dividimos uma entrega de pacotes via *multicast*, da origem até todos os membros do grupo, em duas entregas. A primeira é uma entrega *unicast* da origem até o roteador central, enquanto a segunda é a entrega do roteador central até todos os membros do grupo. Perceba que a primeira parte da entrega precisa ser realizada por meio de encapsulamento. O pacote *multicast* criado pela origem precisa ser encapsulado em um pacote *unicast* e enviado para o roteador central, que desencapsula o pacote *unicast*, extrai o pacote *multicast* e o envia aos membros do grupo. Embora a redução no número de árvores nessa abordagem pareça muito atraente, essa abordagem tem sua própria carga adicional de processamento: o uso de um algoritmo para selecionar um roteador dentre todos os possíveis para atuar como o roteador central de um grupo.

4.4.3 Protocolos de roteamento intradomínio

Nas últimas décadas, surgiram diversos protocolos de roteamento *multicast* intradomínio. Nesta seção, discutimos três deles. Dois são extensões de protocolos de roteamento *unicast* (RIP e OSPF) usando a abordagem de árvores baseadas na origem; o terceiro é um protocolo independente que está se tornando cada vez mais popular. Ele pode ser usado em dois modos, empregando a abordagem de árvores baseadas na origem ou a abordagem de árvores compartilhadas pelo grupo.

Vetor de Distâncias *Multicast*

● **Protocolo de Roteamento *Multicast* por Vetor de Distâncias** (DVMRP – Distance Vector Multicast Routing Protocol) é uma extensão do RIP, usado no roteamento *unicast*. Ele usa a abordagem de árvores baseadas na origem para realizar o *multicast*. Vale ressaltar que, nesse protocolo, cada roteador que recebe um pacote *multicast* para ser encaminhado cria implicitamente uma árvore de *multicast* baseada na origem por meio de três etapas:

1. O roteador usa um algoritmo denominado *Encaminhamento pelo Caminho Reverso* (RPF – Reverse Path Forwarding) para simular a criação de uma parte da árvore ótima baseada na origem entre a origem e ele próprio.
2. O roteador usa um algoritmo denominado *Broadcast pelo Caminho Reverso* (RPB – Reverse Path Broadcasting) para criar uma árvore (de abrangência) de *broadcast* cuja raiz é o próprio roteador e cujas folhas são todas as redes na internet.
3. O roteador usa um algoritmo denominado *Multicast pelo Caminho Reverso* (RPM – Reverse Path Multicasting) para criar uma árvore de *multicast*, cortando alguns galhos da árvore que terminam em redes sem membros do grupo.

Encaminhamento pelo Caminho Reverso

O primeiro algoritmo, o **Encaminhamento pelo Caminho Reverso** (RPF – Reverse Path Forwarding), força o roteador a encaminhar um pacote *multicast* por uma interface específica: aquela que corresponde ao caminho mais curto da origem até o roteador. Como um roteador sabe qual interface está nesse caminho se o roteador não tem uma árvore de menor caminho com a origem na raiz? O roteador usa a primeira propriedade da árvore de menor caminho que discutimos no roteamento *unicast*, a qual diz que o caminho mais curto de A para B é também o caminho mais curto de B para A. O roteador não conhece o caminho mais curto da origem para si próprio, mas pode descobrir qual é o próximo roteador no caminho mais curto dele próprio até a origem (caminho reverso). O roteador simplesmente consulta sua tabela de roteamento *unicast*, como se quisesse enviar um pacote para a origem; a tabela de roteamento fornece o próximo roteador e a interface pela qual o pacote deve ser enviado na direção

reversa. O roteador usa essas informações para aceitar um pacote *multicast* somente se ele chegar por essa interface. Isto é necessário para evitar a criação de laços. No *multicast*, um pacote pode chegar ao mesmo roteador que o tenha transmitido. Se o roteador não descartar todos os pacotes entrantes exceto pelo primeiro, múltiplas cópias do pacote ficarão circulando na internet. Obviamente, o roteador pode adicionar um rótulo no pacote quando chega pela primeira vez e descartar os pacotes que chegarem com esse rótulo, mas a estratégia do RPF é mais simples.

Broadcast pelo Caminho Reverso (RPB)

O algoritmo RPF ajuda um roteador a encaminhar apenas uma cópia recebida de uma origem e descartar as restantes. No entanto, quando ponderamos sobre o *broadcast* na segunda etapa, precisamos lembrar que os destinos são todas as redes (LANs) na internet. Para garantir eficiência, devemos evitar que cada rede receba mais de uma cópia do pacote. Se uma rede estiver conectada a mais de um roteador, ela pode receber uma cópia do pacote de cada roteador. O RPF não pode ajudar nesse caso, porque uma rede não tem a inteligência necessária para aplicar o algoritmo RPF; precisamos permitir que somente um dos roteadores conectados a uma rede encaminhe o pacote para a rede. Uma maneira de fazer isso é designar apenas um roteador como o *pai* (*parent*) de uma rede com relação a uma origem específica. Quando um roteador que não é o pai da rede à qual está conectado recebe um pacote *multicast*, ele simplesmente descarta o pacote. Existem várias formas pelas quais o pai da rede com relação a uma origem pode ser selecionado; uma forma é selecionar o roteador que tem o caminho mais curto até a origem (usando a tabela de roteamento *unicast*, mais uma vez no sentido reverso). Se houver um empate nesse critério, o roteador com o menor endereço IP pode ser selecionado. O leitor deve ter notado que o RPB de fato cria uma árvore de *broadcast* a partir do grafo gerado pelo algoritmo RPF. O RPB corta os galhos da árvore responsáveis por criar ciclos no grafo. Se usarmos o critério do caminho mais curto para escolher o roteador pai, na verdade, estaremos criando uma árvore de *broadcast* de menor custo. Em outras palavras, após esse passo, temos uma árvore de menor custo na qual a origem é a raiz e todas as redes (LANs) são as folhas. Todos os pacotes provenientes da origem atingem todas as LANs na internet atravessando o caminho mais curto. A Figura 4.96 mostra como o RPB pode evitar recepções duplicadas em uma rede por meio da indicação de um roteador pai, R1, designado para a rede N.

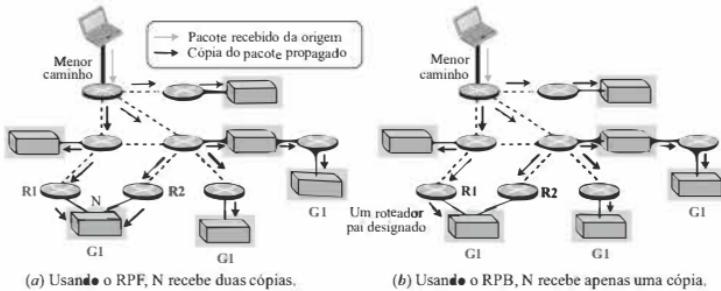



Figura 4.96 RPF *versus* RPB.

Multicast pelo Caminho Reverso

Como você deve ter notado, o RPB não envia o pacote via *multicast*, mas, sim, via *broadcast*, o que não é eficiente. Para aumentar a eficiência, o pacote transmitido via *multicast* deve chegar apenas às redes que têm membros ativos daquele grupo específico. Isto é denominado ***Multicast* pelo**

Caminho Reverso (RPM – Reverse Path Multicasting). Para transformar a árvore de *broadcast* de menor custo em uma árvore de *multicast* de menor custo, cada roteador precisa podar (desativar) as interfaces que não chegam a uma rede com membros ativos correspondentes a uma combinação particular de origem-grupo. Essa etapa pode ser feita de baixo para cima, partindo das folhas até a raiz. No nível das folhas, os roteadores conectados à rede coletam informações sobre participação em grupos utilizando o protocolo IGMP discutido anteriormente. O roteador pai da rede pode, então, distribuir essa informação usando a árvore de menor custo reversa indo do roteador até a origem, da mesma forma que as mensagens de vetor de distâncias são passadas de um vizinho para o outro. Quando um roteador recebe todas essas mensagens relacionadas à participação em grupos, ele tem condições de saber quais interfaces precisam ser podadas. Obviamente, como esses pacotes são distribuídos periodicamente, se novos membros são adicionados a algumas redes, todos os roteadores são informados e podem alterar o estado de suas interfaces de acordo. Entradas e saídas são tratadas continuamente. A Figura 4.97 mostra como a poda no RPM faz com que apenas as redes com membros ativos de um grupo recebam uma cópia do pacote, exceto pelas redes que estão no caminho para uma rede com um membro ativo (que também recebem o pacote).

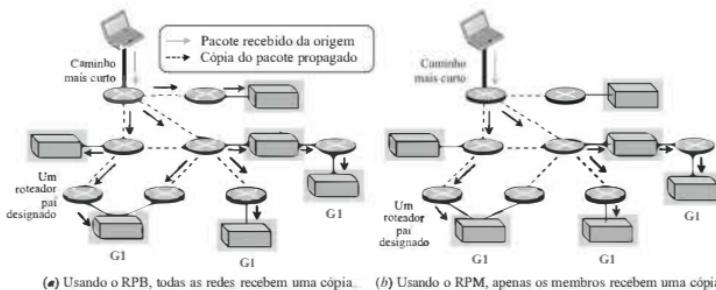



Figura 4.97 RPB *versus* RPM.

Estado de Enlace *Multicast*

O **Protocolo *Multicast* Aberto de Menor Rota Primeiro** (MOSPF – Multicast Open Shortest Path First) é uma extensão do protocolo OSPF, usado no roteamento *unicast*. O MOSPF também usa a abordagem de árvores baseadas na origem para fazer o *multicast*. Se a internet estiver executando um algoritmo de roteamento *unicast* por estado de enlace, a ideia pode ser estendida para criar um algoritmo de roteamento *multicast* por estado de enlace. Lembre-se que no roteamento *unicast* por estado de enlace, cada roteador na internet tem uma Base de Dados de Estado de Enlace (LSDB) que pode ser usada para criar uma árvore de menor custo. Para estender o *unicast* para o *multicast*, cada roteador precisa de outra base de dados, assim como no roteamento *unicast* por vetor de distâncias, para identificar a interface que tem um membro ativo de um grupo em particular. O roteador executa, então, os seguintes passos para encaminhar um pacote *multicast* recebido de uma origem O e que deve ser enviado para um destino G (um grupo de destinatários):

1. O roteador usa o algoritmo de Dijkstra para criar uma árvore de menor custo tendo O como raiz e todos os destinos na internet como folhas. Perceba que essa árvore de menor custo é diferente daquela que o roteador normalmente utiliza para o encaminhamento *unicast*, pois nesta, a raiz da árvore é o próprio roteador. Nesse caso, a raiz da árvore é a origem do pacote especificada no endereço de origem do pacote. O roteador é capaz de criar essa árvore porque ele possui a LSDB, a qual representa a topologia de toda a internet;

o algoritmo de Dijkstra pode ser usado para criar uma árvore com qualquer raiz, não importando qual roteador está aplicando o algoritmo. O ponto que precisamos lembrar é que a árvore de menor custo criada dessa forma depende da origem específica: para cada origem, precisamos criar uma árvore diferente.

2. O roteador encontra a si mesmo na árvore de caminho mais curto criada no primeiro passo. Em outras palavras, o roteador cria uma subárvore de menor custo na qual ele próprio é a raiz.
3. A subárvore de menor custo é, na verdade, uma subárvore de *broadcast* cuja raiz é o roteador e cujas folhas são todas as redes. O roteador utiliza, então, uma estratégia semelhante à que descrevemos no caso do DVMRP para podar a árvore de *broadcast* e transformá-la em uma árvore de *multicast*. O protocolo IGMP é usado para localizar as informações no nível das folhas. O MOSPF introduziu um novo tipo de pacote de atualização de estado de enlace que envia as informações de participação em grupos para todos os roteadores via inundação. O roteador pode usar as informações recebidas dessa forma e podar a árvore de *broadcast* de acordo, transformando-a em uma árvore de *multicast*.
4. O roteador pode, então, encaminhar o pacote recebido por apenas um das interfaces correspondentes aos ramos da árvore de *multicast*. Precisamos garantir que uma cópia do pacote *multicast* atinja todas as redes que apresentam membros ativos do grupo e, ao mesmo tempo, que ela não atinja redes nas quais não há membros ativos.

A Figura 4.98 mostra um exemplo de como usar os passos anteriores para transformar um grafo em uma árvore de *multicast*. Não mostramos a rede para facilitar, mas adicionamos os grupos a cada roteador. A figura mostra como uma árvore baseada na origem é criada com a origem como raiz e transformada em uma subárvore de *multicast* na qual a raiz é o roteador corrente.

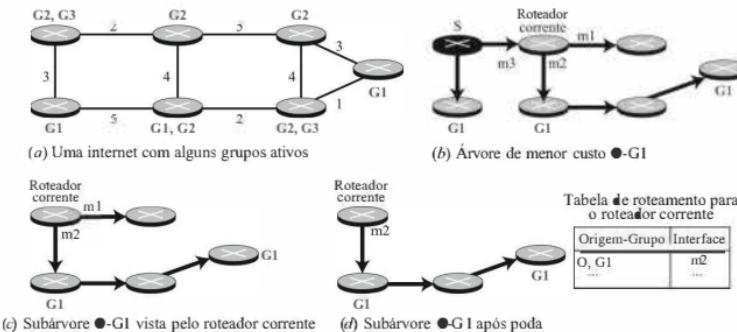



Figura 4.98 Exemplo de formação de árvores no MOSPF.

Multicast Independente de Protocolo

***Multicast* Independente de Protocolo** (PIM – Protocol Independent Multicast) é o nome dado a um protocolo comum que requer um protocolo de roteamento *unicast* para sua operação, porém este pode ser tanto baseado em vetor de distâncias como em estado de enlace. Em outras palavras, o PIM precisa usar a tabela de roteamento de um protocolo de roteamento *unicast* para determinar o próximo roteador em um caminho para o destino, mas não lhe importa como a tabela de roteamento é criada. O PIM tem uma outra característica interessante, que é o fato de poder operar em dois modos

distintos: denso e esparso. O termo *denso*, nesse caso, significa que o número de membros ativos de um grupo na internet é grande, de modo que a probabilidade de um roteador ter um membro de um grupo é alta. Isto pode acontecer, por exemplo, em uma teleconferência popular com muitos membros. O termo *esparso*, por outro lado, significa que apenas alguns roteadores na internet têm membros ativos do grupo, de modo que a probabilidade de que um roteador possua um membro do grupo é baixa. Isto pode acontecer, por exemplo, em uma teleconferência muito técnica, na qual os poucos membros estejam distribuídos pela internet. Quando o protocolo está operando no modo denso, nos referimos a ele como PIM-DM; quando está operando no modo esparso, é denominado PIM-SM. Explicaremos os dois modos de operação a seguir.

Multicast Independente de Protocolo – Modo Denso (PIM-DM)

Quando o número de roteadores com membros conectados é grande em relação ao número de roteadores na internet, o PIM opera no modo denso e é denominado *PIM-DM*. Nesse modo, o protocolo utiliza uma abordagem de árvores baseadas na origem e opera de modo semelhante ao DVMRP, porém de forma mais simples. O PIM-DM emprega apenas duas estratégias descritas no DVMRP: RPF e RPM. Porém, ao contrário do DVMRP, o encaminhamento de um pacote não é suspenso até que ocorra a poda da primeira subárvore. Explicaremos os dois passos utilizados no PIM-DM para esclarecer esse assunto.

1. Um roteador que recebeu um pacote *multicast* vindo da origem O e destinado para o grupo G, primeiramente usa a estratégia do RPF para evitar a recepção de um pacote duplicado. Ele consulta a tabela de roteamento do protocolo *unicast* subjacente para determinar o próximo roteador se quiser enviar uma mensagem para a origem O (no sentido reverso). Se o pacote não tiver vindo do próximo roteador na direção reversa, o roteador descarta o pacote e envia uma mensagem de poda naquela direção para evitar a recepção de futuros pacotes relativos a (O, G).
2. Se o pacote no primeiro passo veio do próximo roteador no sentido reverso, o roteador receptor encaminha o pacote por todas as suas interfaces, exceto por aquela de onde o pacote chegou e pelas interfaces pelas quais ele já recebeu alguma mensagem de poda relativa a (O, G). Perceba que essa transmissão é de fato feita via *broadcast* em vez de *multicast* caso o pacote seja o primeiro vindo da origem O para o grupo G. No entanto, cada roteador no sentido do fluxo que venha a receber um pacote indesejado envia uma mensagem de poda para o roteador no sentido inverso ao fluxo e, eventualmente, o *broadcast* é substituído pelo *multicast*. Note que o DVMRP se comporta de forma diferente: ele requer que as mensagens de poda (que são parte dos pacotes DV) cheguem, de modo que a árvore é podada antes que qualquer mensagem seja enviada via interfaces não podadas. O PIM-DM não se preocupa em tomar essa precaução, porque ele assume que a maioria dos roteadores tem interesse no grupo (a ideia por trás do modo denso).

A Figura 4.99 mostra a ideia por trás do PIM-DM. O primeiro pacote é transmitido a todas as redes, que podem ou não ter membros interessados no grupo G1. Depois que uma mensagem de poda proveniente de um roteador sem membros ativos chega, o segundo pacote é enviado apenas via *multicast*.

Multicast Independente de Protocolo – Modo Esparso (PIM-SM)

Quando o número de roteadores com membros ativos conectados a eles é pequeno em relação ao número de roteadores na internet, o PIM opera no modo esparso e é denominado *PIM-SM*. Nesse cenário, o uso de um protocolo que transmita os pacotes via *broadcast* até que a árvore seja podada não se justifica; o PIM-SM emprega a abordagem de árvore compartilhada pelo grupo para realizar o *multicast*. O roteador central no PIM-SM é chamado ponto de encontro (RP – rendezvous point).

(*a*) Primeiro pacote é transmitido via *broadcast*. (*b*) Segundo pacote é transmitido via *multicast*.

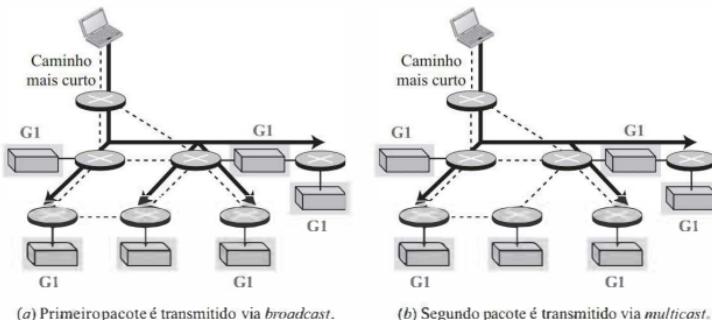



Figura 4.99 Ideia por trás do PIM-DM.

A comunicação *multicast* é realizada em dois passos. Qualquer roteador que tenha um pacote *multicast* para enviar a um grupo de destinos começa encapsulando o pacote *multicast* em um pacote *unicast* (tunelamento) e o envia para o RP. O RP, então, desencapsula o pacote *unicast* e envia o pacote *multicast* para seus destinos.

O PIM-SM utiliza um algoritmo complexo para selecionar um roteador dentre todos os roteadores na internet como o RP para um grupo específico. Isto significa que, se tivermos *m* grupos ativos, precisaremos de *m* RPs, embora um roteador possa servir mais de um grupo. Após a seleção do RP de cada grupo, cada roteador cria uma base de dados na qual ele armazena o identificador do grupo e o respectivo endereço IP do RP, podendo, então, fazer o tunelamento de pacotes *multicast* para ele.

O PIM-SM usa uma árvore de abrangência (*spanning tree*) de *multicast* cuja raiz é o RP e cujas folhas são ponteiros para os roteadores designados conectados a cada rede na qual haja um membro ativo. Um ponto muito interessante sobre o PIM-SM refere-se á formação da árvore de *multicast* para um grupo. A ideia é que cada roteador ajude a criar a árvore. O roteador deve conhecer a única interface a partir da qual ele deve aceitar um pacote *multicast* destinado a um grupo (algo feito pelo RPF no DVMRP). O roteador deve também conhecer a interface (ou interfaces) pela qual ele deve enviar um pacote *multicast* destinado a um grupo (algo feito pelo RPM no DVMRP). Para evitar a entrega de mais de uma cópia do mesmo pacote a uma rede através de vários roteadores (algo conseguido usando o RPB no DVMRP), o PIM-SM requer que os roteadores designados enviem apenas mensagens PIM-SM, como veremos mais adiante.

Para criar uma árvore *multicast* enraizada no RP, o PIM-SM usa mensagens de associação e de poda. A Figura 4.100 mostra a operação das mensagens de associação e poda no PIM-SM. Primeiramente, as três redes associam-se ao grupo G1 e formam uma árvore *multicast*. Mais tarde, uma das redes abandona o grupo e a árvore é podada.

A mensagem de associação é usada para adicionar possíveis novos ramos à árvore, enquanto a mensagem de poda é usada para cortar ramos desnecessários. Quando um roteador designado descobre que uma rede tem um novo membro no grupo correspondente (via IGMP), ele envia uma mensagem de associação na forma de um pacote *unicast* destinado ao RP. O pacote é roteado de acordo com a árvore *unicast* de menor custo para chegar ao RP. Qualquer roteador no caminho recebe e encaminha o pacote, mas, ao mesmo tempo, o roteador adiciona duas informações à sua tabela de roteamento *multicast*. O número da interface pela qual a mensagem de associação chegou é marcado (caso já não o esteja) como uma das interfaces pelas quais o pacote *multicast* destinado ao grupo deve ser enviado no futuro. O roteador também adiciona um contador para o número de

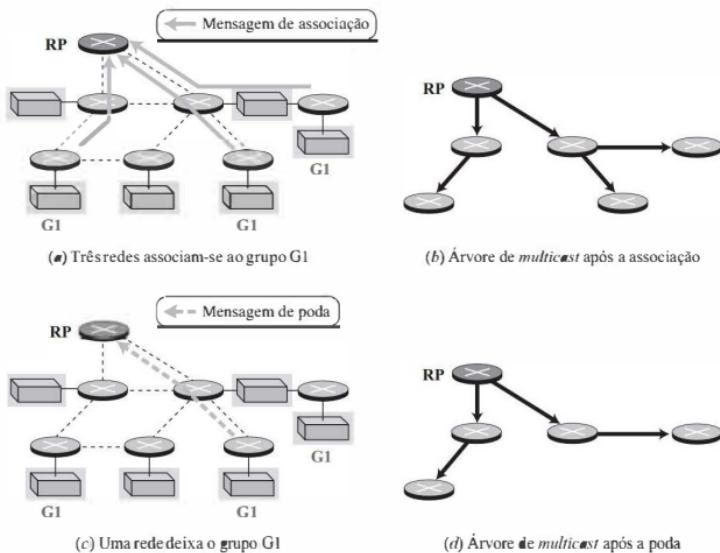



Figura 4.100 Ideia por trás do PIM-SM.

mensagens de associação recebidas por aquela interface, conforme discutiremos em breve. O número da interface pela qual a mensagem de associação foi enviada ao RP é marcado (caso já não o esteja) como a única interface pela qual o pacote *multicast* destinado ao mesmo grupo deve ser recebido. Assim, a primeira mensagem de associação enviada por um roteador designado cria um caminho do RP até uma das redes com membros ativos do grupo.

Para evitar o envio de pacotes *multicast* a redes sem membros ativos, o PIM-SM usa a mensagem de poda. Quando um roteador designado descobre (via IGMP) que não há membros ativos em sua rede, ele envia uma mensagem de poda ao RP. Quando um roteador recebe uma mensagem de poda, decrementa a contagem de associações para a interface pela qual a mensagem chegou e encaminha a mensagem para o roteador seguinte. Quando a contagem de associações para uma interface atinge o valor de zero, aquela interface deixa de fazer parte da árvore de *multicast*.

4.4.4 Protocolos de roteamento interdomínios

Os três protocolos que discutimos para roteamento *multicast*, DVMRP, MOSPF e PIM, foram projetados para fornecer comunicação *multicast* dentro de um sistema autônomo. Quando os membros dos grupos encontram-se distribuídos por diferentes domínios (AS), precisamos de um protocolo de roteamento *multicast* interdomínios.

Um protocolo comum para realizar o roteamento *multicast* interdomínios é o chamado *Protocolo de Roteador de Borda Multicast* (MBGP – Multicast Border Gateway Protocol), que é a extensão do protocolo BGP para roteamento *unicast* interdomínios que discutimos anteriormente. O MBGP fornece dois caminhos entre sistemas autônomos: um para comunicação *unicast* e outro para *multicast*. As informações relativas ao *multicast* são trocadas entre

roteadores de borda em AS diferentes. O MBGP é um protocolo de roteamento baseado em árvore compartilhada pelo grupo no qual um roteador em cada AS é escolhido como o ponto de encontro (RP – rendezvous point).

O problema com o protocolo MBGP é que é difícil informar um RP sobre as origens de grupos *multicast* localizadas em outros AS. O Protocolo *Multicast* de Descoberta de Origem (MSDP – Multicast Source Discovery Protocol) é um protocolo recentemente sugerido para atacar esse problema. Ele atribui a um roteador em cada AS a tarefa de atuar como o representante da origem, informando todos os RPs sobre a existência de origens de dados *multicast* naquele AS.

Outro protocolo considerado um possível substituto para o MBGP é o Protocolo *Multicast* de Roteador de Borda (BGMP – Border Gateway Multicast Protocol), que permite a construção de uma árvore compartilhada pelo grupo tendo uma única raiz em um dos AS. Em outras palavras, para cada grupo, existe apenas uma árvore compartilhada com folhas em AS diferentes, cuja raiz fica localizada em um dos AS. Os dois problemas envolvidos nesse caso, obviamente, são como designar um AS como a raiz da árvore e como informar todas as origens de dados *multicast* sobre a localização da raiz para a qual elas devem tunelar seus pacotes *multicast*.

4.5 NOVA GERAÇÃO DO IP

Na última seção deste capítulo, discutimos a nova geração do IP, o IPv6. O esgotamento de endereços IPv4 e outras deficiências desse protocolo levaram à criação de uma nova versão do protocolo IP no início de 1990. A nova versão, chamada **Protocolo Internet versão 6 (IPv6)** ou **nova geração do IP (IPng)** é uma proposta que visa aumentar o espaço de endereços do IPv4 e, ao mesmo tempo, reformular o formato do pacote IP e revisar alguns protocolos auxiliares, como o ICMP. É interessante saber que o IPv5 era uma proposta, baseada no modelo OSI, que nunca se materializou. A seguir, mostramos as principais mudanças no protocolo IPv6:

- **Maior espaço de endereços.** Um endereço IPv6 apresenta um comprimento de 128 *bits*. Comparado com o endereço de 32 *bits* do IPv4, é uma expansão enorme (2^{96} vezes) no espaço de endereços.
- **Formato de cabeçalho mais adequado.** O IPv6 utiliza um novo formato de cabeçalho, no qual as opções são separadas do cabeçalho-base e inseridas, quando necessário, entre o cabeçalho-base e os dados. Isso simplifica e acelera o processo de roteamento, pois a maioria das opções não precisa ser verificada pelos roteadores.
- **Novas opções.** O IPv6 apresenta novas opções para dar suporte a funcionalidades adicionais.
- **Suporte a extensões.** O IPv6 foi projetado para permitir a extensão do protocolo caso necessário por novas tecnologias ou aplicações.
- **Suporte a alocação de recursos.** No IPv6, o campo de tipo de serviço foi removido, mas dois novos campos, classe de tráfego e rótulo de fluxo, foram adicionados para permitir que a fonte solicite um tratamento especial para o pacote. Esse mecanismo pode ser utilizado para fornecer suporte a classes de tráfego, como vídeo e áudio em tempo real.
- **Suporte à maior segurança.** As opções de cifração e autenticação no IPv6 fornecem confidencialidade e integridade ao pacote.

A adoção do IPv6 tem sido lenta, pois a motivação original para o seu desenvolvimento, o esgotamento de endereços IPv4, tem sido atrasada em razão de três métodos paliativos: o endereçamento sem classes, o uso de DHCP para atribuição dinâmica de endereços e o NAT. Entretanto, a rápida disseminação do uso da Internet, bem como o surgimento de novos serviços, como IP móvel, telefonia IP e IP com suporte à telefonia móvel, poderá exigir a substituição total do IPv4 pelo IPv6.

Previa-se que todos os *hosts* do mundo estariam usando IPv6 em 2010, mas isso não aconteceu. A previsão atual é de que isso ocorra em 2020*.

Nesta seção, começamos discutindo o formato do pacote IPv6. Então, abordamos o endereçamento IPv6, que adota uma estrutura um pouco diferente da estrutura da versão 4. A seguir, mostraremos como realizar a difícil tarefa de fazer a transição da versão antiga para a nova versão por meio de algumas estratégias recomendadas. Finalmente, discutiremos o ICMPv6, o único protocolo auxiliar na camada de rede, substituindo vários protocolos existentes na versão 4.

4.5.1 Formato do pacote

O pacote IPv6 é mostrado na Figura 4.101. Cada pacote é composto por um cabeçalho-base, que é seguido pela carga útil, ou ***payload***. O cabeçalho-base ocupa 40 *bytes*, enquanto a carga útil pode ser de até 65.535 *bytes* de informação. A seguir, apresentamos a descrição dos campos.

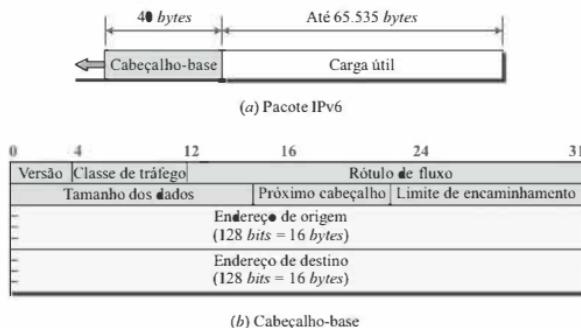



Figura 4.101 Datagrama IPv6.

- **Versão.** O campo de versão de 4 *bits* define o número da versão do IP. Para o IPv6, o valor é 6.
- **Classe de tráfego.** O campo de classe de tráfego de 8 *bits* é usado para distinguir diferentes cargas úteis com diferentes requisitos para sua entrega. Ele substitui o campo *tipo de serviço* do IPv4.
- **Rótulo de fluxo.** O rótulo de fluxo é um campo de 20 *bits* concebido para proporcionar tratamento especial para um fluxo de dados em particular. Discutiremos esse campo mais adiante.
- **Tamanho dos dados.** O campo de tamanho dos dados de 2 *bytes* define o comprimento do datagrama IP excluindo-se o cabeçalho. Perceba que o IPv4 define dois campos relacionados a comprimento: comprimento do cabeçalho e comprimento total. No IPv6, o comprimento do cabeçalho-base é fixo (40 *bytes*); portanto, apenas o comprimento da carga útil precisa ser definido.

---

* N. de T.: A previsão do CGI.br (Comitê Gestor da Internet no Brasil) é de que o esgotamento do estoque de endereços IPv4 aconteça até 2014. Mais informações sobre a adoção do IPv6 no Brasil podem ser encontradas em http://www.ipv6.br.

- **Próximo cabeçalho.** O próximo cabeçalho é um campo de 8 *bits* que define o tipo do primeiro cabeçalho de extensão (caso ele esteja presente) ou o tipo dos dados que se seguem ao cabeçalho-base no datagrama. Esse campo é similar ao campo de protocolo no IPv4, mas discutiremos isso mais profundamente quando abordarmos a carga útil.
- **Limite de encaminhamento.** O campo de limite de encaminhamento de 8 *bits* tem a mesma finalidade que o campo TTL do IPv4.
- **Endereços de origem e destino.** O campo de endereço de origem consiste em um endereço de Internet de 16 *bytes* (128 *bits*) que identifica a fonte original do datagrama. O campo de endereço de destino consiste em um endereço de Internet de 16 *bytes* (128 *bits*) que identifica o destino do datagrama.
- **Carga útil.** Em comparação com o IPv4, o campo de carga útil (*payload*) no IPv6 tem um formato e significado diferentes, conforme mostra a Figura 4.102.

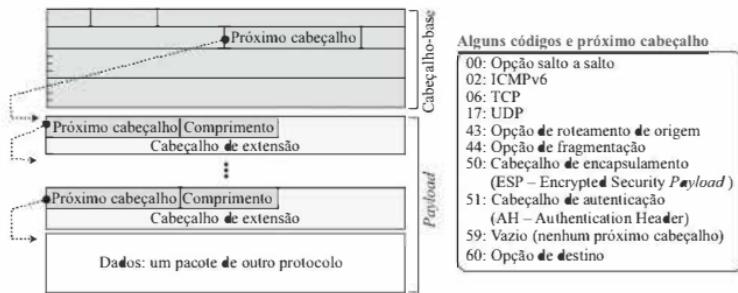



Figura 4.102 Carga útil em um datagrama IPv6.

A carga útil no IPv6 consiste em uma combinação de zero ou mais cabeçalhos de extensão (opções) seguidos pelos dados de outros protocolos (UDP, TCP, e assim por diante). No IPv6, as opções, que fazem parte do cabeçalho do IPv4, foram projetadas como cabeçalhos de extensão. A carga útil pode ter quantos cabeçalhos de extensão forem necessários, conforme a situação. Cada cabeçalho de extensão apresenta dois campos obrigatórios, o próximo cabeçalho e o comprimento, seguidos de informações relacionadas àquela opção em particular. Perceba que cada valor (código) do campo de próximo cabeçalho define o tipo do próximo cabeçalho (opção salto a salto, opção de roteamento de origem, etc.); o último campo de próximo cabeçalho especifica o protocolo (UDP, TCP, etc.) que é transportado pelo datagrama.

Conceito de fluxo e prioridade no IPv6

O protocolo IP foi originalmente concebido como um protocolo não orientado à conexão. Entretanto, a tendência atual é usar o protocolo IP como um protocolo orientado à conexão. A tecnologia MPLS descrita anteriormente nos permite encapsular um pacote IPv4 em um cabeçalho MPLS usando um campo de rótulo, ou identificador. Na versão 6, o rótulo de fluxo foi adicionado diretamente ao formato do datagrama IPv6 para permitir o uso do IPv6 como um protocolo orientado à conexão.

Para um roteador, um fluxo é uma sequência de pacotes que compartilham as mesmas características, tais como o fato de viajarem pelo mesmo caminho, usarem os mesmos recursos, apresentarem o mesmo tipo de segurança e assim por diante. Um roteador que suporta o processamento de rótulos de fluxo tem uma tabela deles. Tal tabela apresenta uma entrada para cada rótulo de fluxo ativo; cada entrada define os serviços exigidos pelo rótulo de fluxo correspondente. Quando

o roteador recebe um pacote, ele consulta sua tabela de rótulos de fluxo para determinar a entrada correspondente ao valor do rótulo de fluxo especificado no pacote, e fornece ao pacote os serviços mencionados na entrada da tabela. No entanto, perceba que o rótulo de fluxo em si não fornece as informações para as entradas da tabela; essa informação deve ser fornecida por outros meios, tais como pela opção salto a salto ou por outros protocolos.

Em sua forma mais simples, um rótulo de fluxo pode ser usado para acelerar o processamento de um pacote por um roteador. Quando um roteador recebe um pacote, em vez de consultar a tabela de roteamento e executar um algoritmo de roteamento para determinar o endereço do próximo salto, ele pode facilmente procurar o próximo salto em uma tabela de rótulos de fluxo.

Em sua forma mais sofisticada, um rótulo de fluxo pode ser utilizado para dar suporte a transmissões de áudio e vídeo em tempo real, que, particularmente na forma digital, necessitam de recursos, como elevada largura de banda e *buffers* de grandes dimensões, apresentam longo tempo de processamento, entre outras características. Um processo pode exigir a reserva desses recursos com antecedência com o intuito de garantir que os dados em tempo real não apresentem atrasos devido à falta de recursos. O uso de dados em tempo real e da reserva desses recursos requer outros protocolos, tais como o Protocolo de Transporte de Tempo Real (RTP – Real-Time Transport Protocol) e o Protocolo de Reserva de Recursos (RSVP – Resource Reservation Protocol), além do IPv6 (ver Capítulo 8).

Fragmentação e remontagem

Ainda há fragmentação e remontagem de datagramas do protocolo IPv6, mas foi introduzida uma grande mudança a esse respeito. Os datagramas IPv6 podem ser fragmentados apenas pela origem, não pelos roteadores; a remontagem acontece no destino. A fragmentação de pacotes nos roteadores não é permitida com o objetivo de acelerar o processamento de pacotes no roteador. A fragmentação de um pacote em um roteador exige muito processamento; os pacotes precisam ser fragmentados e todos os campos relacionados à fragmentação devem ser recalculados. No IPv6, a origem pode verificar o tamanho do pacote e tomar a decisão de fragmentá-lo ou não. Quando um roteador recebe o pacote, ele pode verificar seu tamanho e descartá-lo se for maior que o permitido pelo MTU da rede adiante. Para informar a origem do ocorrido, o roteador envia, então, uma mensagem de relatório de erro do tipo "pacote muito grande" usando o ICMPv6 (discutido mais adiante).

Cabeçalhos de extensão

Os cabeçalhos de extensão têm um papel importante e necessário no IPv6. Em particular, três cabeçalhos de extensão – fragmentação, cabeçalho de autenticação e cabeçalho de encapsulamento – estão presentes em alguns pacotes. Para encurtar este capítulo, removemos daqui a discussão sobre cabeçalhos de extensão, mas ela pode ser encontrada no *site* do livro como parte do material extra para o Capítulo 4.

4.5.2 Endereçamento IPv6

A principal razão para a migração do IPv4 para o IPv6 é o reduzido tamanho do espaço de endereços no IPv4. Nesta seção, mostramos como o enorme espaço de endereços IPv6 previne o esgotamento de endereços no futuro. Discutimos também como o novo endereçamento resolve alguns problemas existentes no mecanismo de endereçamento do IPv4. Um endereço IPv6 tem 128 *bits* ou 16 *bytes* (octetos) de comprimento, quatro vezes o comprimento de endereços no IPv4.

Um computador normalmente armazena o endereço em binário, mas é claro que 128 *bits* não podem ser facilmente manipulados por seres humanos. Diversas notações foram propostas para representar os endereços IPv6 quando eles são manipulados por seres humanos. A seguir, mostramos duas delas: a binária e a hexadecimal com dois pontos.

Binária	**1111111011110110 ... 1111111100000000**
Hexadecimal com dois pontos	**FEF6:BA98:7654:3210:ADEF:BBFF:2922:FF00**

A notação binária é utilizada quando os endereços são armazenados em um computador. A notação hexadecimal com dois pontos divide o endereço em oito seções, cada uma composta por quatro dígitos hexadecimais separados por dois pontos.

Embora um endereço IPv6, mesmo no formato hexadecimal, seja muito longo, muitos dos dígitos costumam ser zeros. Nesse caso, podemos abreviar o endereço. Os zeros iniciais de uma seção podem ser omitidos. Usando essa forma de abreviação, 0074 pode ser escrito como 74, 000F como F e 0000 como 0. Perceba que 3210 não pode ser abreviado. Uma forma de abreviação adicional, comumente denominada compressão de zeros, pode ser aplicada à notação hexadecimal com dois pontos caso haja seções consecutivas compostas apenas por zeros. Nesse caso, podemos remover todos os zeros e substituí-los por um par de dois pontos.

FDEC:0:0:0:0:BBFF:0:FFFF → **FDEC::BBFF:0:FFFF**

É importante ressaltar que esse tipo de abreviação é permitido apenas uma vez por endereço. Se houver mais do que uma sequência de seções compostas apenas por zeros, somente uma delas pode ser comprimida.

Algumas vezes, encontramos uma representação mista de um endereço IPv6: a combinação das notações hexadecimal com dois pontos e decimal com pontos. Isto é apropriado durante o período de transição no qual um endereço IPv4 encontra-se incorporado a um endereço IPv6 (ocupando os 32 *bits* mais à direita). Podemos utilizar a notação hexadecimal com dois pontos para as seis seções mais à esquerda e a notação decimal com pontos para os quatro *bytes* das duas seções mais à direita. No entanto, isto normalmente acontece quando todos ou a maioria das seções mais à esquerda do endereço IPv6 são compostas apenas por 0s. Por exemplo, o endereço (**::130.24.24.18**) é um endereço IPv6 legítimo, no qual a compressão de zeros mostra que todos os 96 *bits* mais à esquerda do endereço são zeros.

Como veremos mais adiante, o IPv6 utiliza endereçamento hierárquico. Por essa razão, o IPv6 permite a notação de barra ou CIDR. Por exemplo, a seguir, mostramos como podemos especificar um prefixo de 60 *bits* utilizando o CIDR. Mais tarde, mostraremos como um endereço IPv6 pode ser dividido em um prefixo e um sufixo.

FDEC::BBFF:0:FFFF/60

Espaço de endereços

O espaço de endereços do IPv6 contém 2^{128} endereços; esse espaço corresponde a 2^{96} vezes o espaço de endereços do IPv4 – definitivamente longe do esgotamento de endereços. Numericamente, o tamanho do espaço é

340.282.366.920.938.463.374.607.431.768.211.456.

Para dar uma ideia do número total de endereços, consideramos que apenas 1/64 (quase 2%) dos endereços nesse espaço possam ser atribuídos a pessoas no planeta Terra e o restante seja reservado a propósitos especiais. Consideramos também que o número de pessoas na Terra esteja prestes a se tornar 2^{34} (mais de 16 bilhões). Cada pessoa pode ter 2^{88} endereços para usar. O esgotamento de endereços nessa versão é impossível.

Três tipos de endereços

No IPv6, um endereço de destino pode pertencer a uma de três categorias: *unicast*, *anycast* e *multicast*. Um endereço *unicast* define uma única interface (computador ou roteador). O pacote enviado a um endereço *unicast* será roteado para o destinatário desejado.

Um endereço *anycast* define um grupo de computadores que compartilham um só endereço. Um pacote com um endereço *anycast* é entregue a apenas um membro do grupo, o mais acessível dentre eles. A comunicação *anycast* é usada, por exemplo, quando existem vários servidores que podem responder a uma solicitação, enviada para o servidor mais facilmente acessível. O *hardware* e o *software* criam apenas uma cópia da solicitação, que chega a apenas um dos servidores. O IPv6 não apresenta um bloco alocado para o *anycast*; os endereços são atribuídos a partir do bloco de endereços *unicast*.

Um endereço *multicast* também define um grupo de computadores. No entanto, existe uma diferença entre o *anycast* e o *multicast*. No *anycast*, uma única cópia do pacote é enviada a um dos membros do grupo; já no *multicast*, cada membro do grupo recebe uma cópia. Como veremos em breve, o IPv6 apresenta um bloco de endereços alocado para *multicast*, a partir do qual um endereço é atribuído aos membros do grupo.

É interessante notar que o IPv6 não define o *broadcast*, nem mesmo em uma forma limitada. Como veremos, o IPv6 considera o *broadcast* um caso especial de *multicast*.

Alocação do espaço de endereços

Assim como o espaço de endereços do IPv4, o espaço de endereços do IPv6 é dividido em vários blocos de tamanhos variados e cada bloco é alocado para um propósito especial. A maioria dos blocos ainda não tem uma atribuição específica, sendo reservados para utilizações futuras. A Tabela 4.7 mostra apenas os blocos com atribuições específicas. Nela, a última coluna mostra a fração que cada bloco ocupa no espaço de endereços completo.

Tabela 4.7 Prefixos para os endereços IPv6 alocados.

Prefixo do bloco	CIDR	Atribuição do bloco	Fração
0000 0000	0000::/8	Endereços especiais	1/256
001	**2000::/3**	***Unicast* global**	**1/8**
1111 110	FC00::/7	*Unicast* único local	1/128
1111 1110 10	FE80::/10	Endereços da rede local	1/1024
1111 1111	FF00::/8	Endereços *multicast*	1/256

Endereços *unicast* globais

O bloco no espaço de endereços usado para comunicações *unicast* (um para um) entre dois *hosts* na Internet é denominado bloco de *endereços unicast globais* (*global unicast addresses*) ou ***endereços roteáveis na Internet***. O CIDR para o bloco é o **2000::/3**, o que significa que os três *bits* mais à esquerda são os mesmos para todos os endereços desse bloco (001). O tamanho dele é 2^{125} *bits*, mais do que suficiente para a expansão da Internet por muitos anos. Um endereço nesse bloco é dividido em três partes: *prefixo de roteamento global* (*n bits*), *identificador de sub-rede* (*m bits*) e *identificador de interface* (*q bits*), conforme mostra a Figura 4.103. A figura também mostra o comprimento recomendado para cada parte.

O prefixo de roteamento global é usado para rotear o pacote através da Internet até uma organização, por exemplo, o ISP, dono do bloco. Como os três primeiros *bits* são fixos (001), o restante dos 45 *bits* pode ser especificado por até 2^{45} locais (uma organização privada ou um ISP). Os roteadores globais na Internet roteiam pacotes para seu local de destino com base no valor de *n*. Os próximos m *bits* (16 *bits* de acordo com a recomendação) definem uma sub-rede dentro de uma organização. Isto significa que uma organização pode ter até 2^{16} = 65.536 sub-redes, o que é mais do que suficiente.

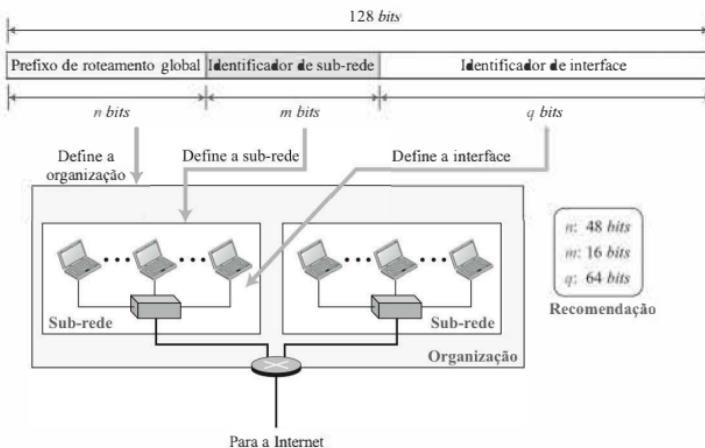



Figura 4.103 Endereços *unicast* globais.

Os últimos *q bits* (64 *bits* de acordo com a recomendação) correspondem ao identificador de interface. O identificador de interface é semelhante ao identificador do nó no endereçamento IPv4, embora o termo identificador de interface seja uma melhor escolha porque, conforme discutimos anteriormente, o identificador do nó, na verdade, define a interface e não o nó. Se o nó for movido de uma interface para outra, seu endereço IP precisa ser alterado.

No endereçamento IPv4, não há uma relação específica entre o identificador do nó (no nível do IP) e o endereço da camada de enlace (na camada de enlace de dados) porque o endereço da camada de enlace é normalmente muito maior do que o identificador do nó. O endereçamento IPv6 permite essa relação. Um endereço da camada de enlace, cujo comprimento seja inferior a 64 *bits* pode ser incorporado inteira ou parcialmente ao identificador de interface, eliminando o processo de mapeamento. Dois esquemas de endereçamento da camada de enlace comuns podem ser utilizados para esse propósito: o Identificador Único Estendido de 64 *bits* (EUI-64 – 64-*bit Extended Unique Identifier*) definido pelo IEEE e o endereço da camada de enlace de 48 *bits* definido pela Ethernet.

Endereços especiais

Depois da discussão sobre o bloco de *unicast* global, discutiremos as características e finalidades dos blocos atribuídos e reservados na primeira linha da Tabela 4.7. Os endereços cujo prefixo é **(0000::/8)** são reservados, mas parte desse bloco é usada para definir alguns endereços especiais. A Figura 4.104 mostra os endereços do bloco com atribuições específicas.

O endereço não especificado é um sub-bloco contendo um único endereço, usado durante a inicialização quando um *host* não sabe seu próprio endereço e quer enviar uma solicitação com o objetivo de determiná-lo (ver seção sobre DHCP).

O endereço de *loopback* também consiste em um único endereço. Discutimos endereços de *loopback* anteriormente no contexto do IPv4. No IPv4, esse bloco consiste em uma faixa de endereços; já no IPv6, o bloco de *loopback* contém um único endereço.

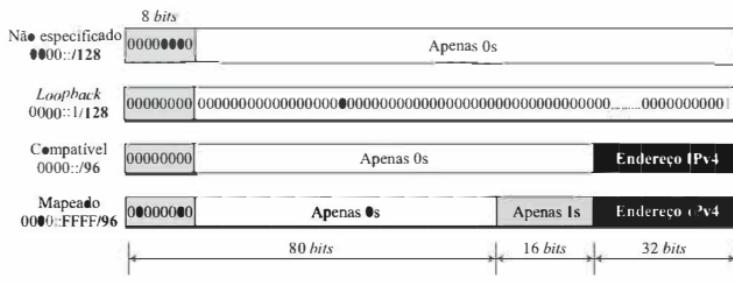



Figura 4.104 Endereços especiais.

Como veremos mais adiante, durante a transição do IPv4 para o IPv6, os *hosts* podem usar os seus endereços IPv4 incorporados aos endereços IPv6. Dois formatos foram projetados para essa finalidade: compatível e mapeado. Um **endereço compatível** é um endereço contendo 96 *bits* de zero seguido pelos 32 *bits* do endereço IPv4. Ele é aplicado quando um computador usando IPv6 deseja enviar uma mensagem para outro computador usando IPv6. Um **endereço mapeado** é usado quando um computador já migrou para o IPv6 e quer enviar uma mensagem para um computador que ainda está usando o IPv4. Um ponto muito interessante sobre endereços mapeados e compatíveis é que eles foram projetados de tal forma que, ao se calcular a soma de verificação (*checksum*), pode-se utilizar tanto o endereço incorporado como o endereço total porque 0s e 1s adicionais localizados em múltiplos de 16 não têm qualquer efeito sobre o cálculo da soma de verificação. Isto é importante para o UDP e o TCP, que utilizam um pseudocabeçalho para calcular a soma de verificação, porque tal cálculo não é afetado se o endereço do pacote for alterado de IPv6 para IPv4 por um roteador.

Outros blocos alocados

O IPv6 utiliza dois grandes blocos para endereçamento privado e um grande bloco para *multicast*, conforme mostra a Figura 4.105. Uma organização pode criar e usar privativamente um sub-bloco em um bloco de *unicast* único local (*unique local unicast*). Um pacote carregando esse tipo de endereço como endereço de destino não deve ser roteado. Esse tipo de endereço apresenta o identificador 1111 110, o próximo *bit* pode ser 0 ou 1 para especificar como o endereço é selecionado (localmente ou por uma autoridade). Os 40 *bits* seguintes são selecionados pela organização usando um número aleatório de 40 *bits*. Isto significa que um total de 48 *bits* define um sub-bloco que se parece com um endereço *unicast* global. O número aleatório de 40 *bits* faz com que a probabilidade de duplicação do endereço seja extremamente pequena. Observe a semelhança entre o formato desses endereços e dos endereços de *unicast* globais. O segundo bloco, projetado para endereços privados, é o bloco de endereços da rede local. Nele, um sub-bloco pode ser usado como um endereço privado em uma rede. Esse tipo de endereço tem como identificador de bloco o valor 1111111010. Os próximos 54 *bits* são fixados em zero. Os últimos 64 *bits* podem ser modificados para definir a interface de cada computador. Observe a semelhança entre o formato desses endereços e do endereço *unicast* global.

Discutimos os endereços *multicast* IPv4 no início do capítulo. Eles são usados para definir um grupo de *hosts* e não apenas um. No IPv6, um grande bloco de endereços foi alocado para o *multicast*. Todos esses endereços utilizam o prefixo 11111111. O segundo campo é um identificador que especifica o endereço do grupo como permanente ou transitório. Um endereço de grupo permanente é definido pelas autoridades da Internet e pode ser acessado a qualquer momento. Um endereço de grupo transitório, por outro lado, é utilizado apenas temporariamente. Sistemas envolvidos em uma teleconferência, por exemplo, podem usar um endereço de grupo transitório. O terceiro campo define o escopo do endereço de grupo. Muitos escopos diferentes foram definidos, conforme mostrado na figura.

Figura 4.105 Bloco de *unicast* único local.

4.5.3 Transição do IPv4 para o IPv6

Embora tenhamos uma nova versão do protocolo IP, como podemos realizar a transição de modo a encerrar o uso do IPv4 e começar a usar o IPv6? A primeira solução que vem à mente é definir uma data de transição na qual cada estação ou roteador deve parar de usar a versão antiga e começar a usar a nova. No entanto, isso não é prático; devido ao grande número de sistemas na Internet, a transição do IPv4 para o IPv6 não pode acontecer de repente. Levará um tempo considerável até que cada sistema na Internet possa passar do IPv4 para o IPv6, e a transição deve ser suave para evitar problemas entre eles. Três estratégias foram criadas pelo IETF para ajudar nessa transição: pilha dupla, tunelamento e tradução de cabeçalho.

Pilha dupla

Recomenda-se que todos os *hosts*, antes de migrar completamente para a versão 6, tenham uma pilha dupla de protocolos durante a transição. Em outras palavras, uma estação deve executar IPv4 e IPv6 simultaneamente até que toda a Internet passe a usar o IPv6. A Figura 4.106 mostra a estrutura de uma configuração de pilha dupla.

Para determinar qual versão do IP deve ser usada ao enviar um pacote para um destino, a estação de origem consulta o DNS. Se o DNS retornar um endereço IPv4, a estação de origem envia um pacote IPv4. Se o DNS retornar um endereço IPv6, a estação de origem envia um pacote IPv6.

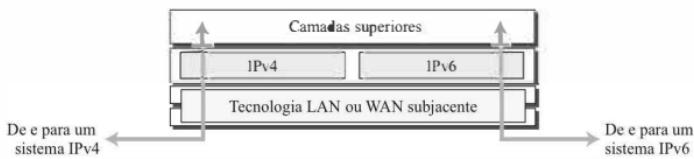



Figura 4.106 Pilha dupla.

Tunelamento

O tunelamento é uma estratégia usada quando dois computadores usando IPv6 desejam se comunicar e o pacote deve atravessar uma região que usa IPv4. Para passar por tal região, o pacote deve ter um endereço IPv4. Assim, o pacote IPv6 é encapsulado em um pacote IPv4 quando entra nessa região e desencapsulado quando sai dela. É como se o pacote IPv6 entrasse em um túnel em uma extremidade e emergisse na outra. Para deixar claro que o pacote IPv4 está carregando um pacote IPv6 como sua carga útil de dados, o valor do campo de protocolo é fixado em 41. O tunelamento é mostrado na Figura 4.107.

Figura 4.107 Estratégia de tunelamento.

Tradução de cabeçalho

A tradução de cabeçalho é necessária quando a maioria da Internet estiver usando o IPv6, mas alguns sistemas ainda estiverem usando o IPv4. O remetente deseja usar o IPv6, mas o destinatário não é capaz de entender IPv6. O tunelamento não funciona nessa situação, porque o pacote deve estar no formato IPv4 para ser compreendido pelo receptor. Nesse caso, o formato do cabeçalho deve ser totalmente alterado por meio da tradução de cabeçalho. O cabeçalho do pacote IPv6 é convertido em um cabeçalho IPv4 (ver Figura 4.108).

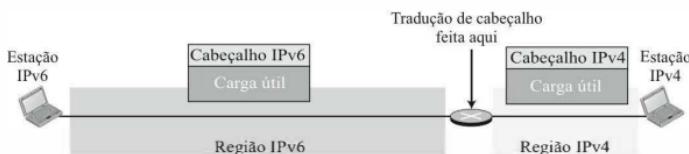



Figura 4.108 Estratégia de tradução de cabeçalho.

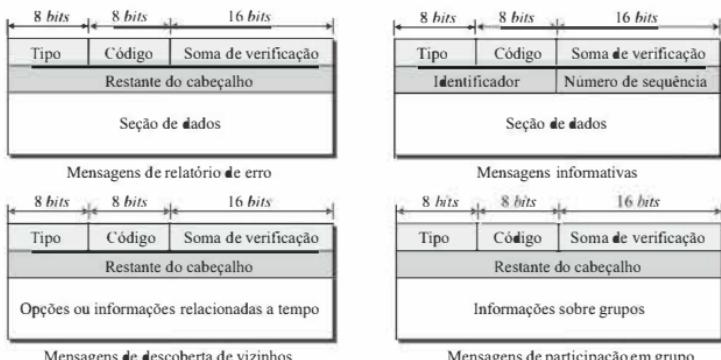
4.5.4 ICMPv6

Outro protocolo modificado na versão 6 da pilha de protocolos TCP/IP é o ICMP. A nova versão, o Protocolo de Mensagens de Controle da Internet versão 6 (ICMPv6 – Internet Control Message Protocol version 6), segue a mesma estratégia e tem os mesmos objetivos da versão 4. O ICMPv6, no entanto, é mais complicado que o ICMPv4: alguns protocolos que eram independentes na versão 4 agora fazem parte do ICMPv6 e algumas novas mensagens foram acrescentadas para torná-lo mais útil. A Figura 4.109 compara a camada de rede da versão 4 com aquela da versão 6. O ICMP, o ARP (discutido no Capítulo 5), e os protocolos IGMP da versão 4 foram combinados em um único protocolo, o ICMPv6.

Figura 4.109 Comparação entre as camadas de rede na versão 4 e na versão 6.

Podemos dividir as mensagens do ICMPv6 em quatro grupos: mensagens de relatório de erro, informativas, de descoberta de vizinhos e de participação em grupo.

As mensagens de relatório de erro seguem a mesma política seguida na versão 4. Existem quatro tipos de mensagens nesse grupo: *destino inalcançável*, *pacote muito grande*, *tempo excedido* e *problema de parâmetro*. A mensagem de *origem extinta* presente na versão 4 foi removida porque ela era raramente utilizada. A mensagem de redirecionamento foi transferida para outra categoria. Uma nova mensagem, *pacote muito grande*, foi adicionada porque, na versão 6, não é permitido a um roteador realizar fragmentação. Se uma mensagem for muito grande, o roteador irá descartá-la e enviará à origem uma mensagem de relatório de erro do tipo pacote muito grande (ver Figura 4.110).

Valor do tipo e do código

Mensagens de relatório de erro
01: Destino inalcançável (códigos 0 a 6)
02: Pacote muito grande (código 0)
03: Tempo excedido (códigos 0 e 1)
04: Problema de parâmetro (códigos 0 a 2)

Mensagens informativas
128 e 129: Pedido e resposta de eco (código 0)

Mensagens de participação em grupo
130: Consulta de participação (código 0)
131: Relatório de participação (código 0)

Mensagens de descoberta de vizinhos
133 e 134: Solicitação e anúncio de roteador (código 0)
135 e 136: Solicitação e anúncio de vizinho (código 0)
137: Redirecionamento (códigos 0 a 3)
141 e 142: Solicitação e anúncio reversos de vizinho (código 0)

Nota: Ver página Web do livro para explicações adicionais sobre essas mensagens.

Figura 4.110 Mensagens ICMPv6.

As mensagens informativas no ICMPv6 desempenham o mesmo papel que as mensagens de consulta no ICMPv4. Há apenas um par de mensagens nesse grupo: mensagens de *pedido de eco* e de *resposta de eco*. Comparando esse grupo com o da versão 4, vemos que as duas mensagens, pedido e resposta de carimbo de tempo (*timestamp*), foram descartadas porque raramente eram usadas.

Um novo grupo, denominado mensagens de descoberta de vizinhos, foi adicionado na versão 6. As mensagens nesse grupo permitem encontrar roteadores em uma vizinhança, encontrar os endereços da camada de enlace de outros *hosts*, encontrar os endereços IPv6 dos *hosts* na vizinhança e redirecionar mensagens. Em outras palavras, esse grupo pode ser dividido em quatro subgrupos. As mensagens de *solicitação de roteador* e *anúncio de roteador* podem ajudar um *host* a encontrar um roteador-padrão para o qual enviar seus pacotes. Na versão 4, o protocolo DHCP era o responsável por tal tarefa. As mensagens de solicitação de vizinho e *anúncio de vizinho* são usadas para tentar encontrar o endereço da camada de enlace de uma estação ou roteador quando o endereço IPv6 é fornecido. Como discutiremos no Capítulo 5, essa tarefa é executada pelo Protocolo de Resolução de Endereços (ARP – Address Resolution Protocol) na versão 4. A mensagem de *redirecionamento* desempenha o mesmo papel que a sua correspondente na versão 4. As mensagens de *solicitação reversa de vizinho* e *anúncio reverso de vizinho* são usadas para determinar o endereço IPv6 de uma estação ou roteador quando o endereço da camada de enlace é fornecido.

Outro novo grupo, o de mensagens de participação em grupo, apresenta duas mensagens: a *consulta de participação* e o *relatório de participação*, que são usadas no *multicast*. Elas desempenham o mesmo papel que as mensagens IGMP descritas anteriormente neste capítulo.

Mais discussões relacionadas ao ICMPv6 podem ser encontradas na página Web do livro, como parte do material extra do Capítulo 4.

4.6 MATERIAL DO FINAL DO CAPÍTULO

4.6.1 Leitura adicional

Livros

Diversos livros fornecem uma cobertura bastante completa dos assuntos discutidos neste capítulo. Recomendamos a leitura de [Com 06], [Tan 03], [Koz 05], [Ste 95], [G & W 04], [Per 00], [Kes 02], [Moy 98], [W & Z 01] e [Los 04].

RFCs

O endereçamento IPv4 é discutido nas RFCs 917, 927, 930, 932, 940, 950, 1122 e 1519. Encaminhamento é discutido nas RFCs 1812, 1971 e 1980. O MPLS é discutido nas RFCs 3031, 3032, 3036 e 3212. O protocolo IPv4 é discutido nas RFCs 791, 815, 894, 1122, 2474 e 2475. O ICMP é discutido nas RFCs 792, 950, 956, 957, 1016, 1122, 1256, 1305 e 1987. O RIP é discutido na RFC 1058 e 2453. O OSPF é discutido nas RFCs 1583 e 2328. O BGP é discutido nas RFCs 1654, 1771, 1773, 1997, 2439, 2918 e 3392. O *multicast* é discutido nas RFCs 1075, 1585, 2189, 2362 e 3376. O endereçamento IPv6 é discutido nas RFCs 2375, 2526, 3513, 3587, 3789 e 4291. O protocolo IPv6 é discutido nas RFCs 2460, 2461 e 2462. O ICMPv6 é discutido nas RFCs 2461, 2894, 3122, 3810, 4443 e 4620.

4.6.2 Termos-chave

- agregação de endereços
- algoritmo de Dijkstra
- Bellman-Ford
- *Broadcast* pelo Caminho Reverso (Reverse Path Broadcasting – RPB)
- comutador *banyan*

- comutador *crossbar*
- correspondência da máscara mais longa
- Encaminhamento pelo Caminho Reverso (Reverse Path Forwarding – RPF)
- endereçamento com classes
- endereçamento sem classes
- endereço *anycast*
- endereço compatível
- endereço de *broadcast*
- endereço de rede
- envenenamento reverso (*poisoned reverse*)
- espaço de endereçamento
- horizonte dividido (*split horizon*)
- *Multicast* Independente de Protocolo (PIM – Protocol Independent Multicast)
- *Multicast* pelo Caminho Reverso (RPM – Reverse Path Multicasting)
- Protocolo Aberto de Menor Rota Primeiro (OSPF – Open Shortest Path First)
- Protocolo *Multicast* Aberto de Menor Rota Primeiro (MOSPF – Multicast Open Shortest Path First)
- Protocolo de Gerenciamento de Grupos Internet (IGMP – Internet Group Management Protocol)
- Protocolo de Informações de Roteamento (RIP – Routing Information Protocol)
- Protocolo de Mensagens de Controle da Internet (ICMP – Internet Control Message Protocol)
- Protocolo de Roteamento de Borda (BGP – Border Gateway Protocol)
- Protocolo de Roteamento *Multicast* por Vetor de Distâncias (DVMRP – Distance Vector Multicast Routing Protocol)
- Roteamento Interdomínios sem Classes (CIDR – Classless Interdomain Routing)
- roteamento por estado de enlace
- roteamento por vetor de caminhos
- roteamento por vetor de distâncias
- Sistema Autônomo (AS – Autonomous System)
- Tradução de Endereços de Rede (NAT – Network Address Translation)
- Unidade Máxima de Transferência (MTU – Maximum Transfer Unit)

4.6.3 Resumo

A Internet é composta por muitas redes conectadas por meio de dispositivos de conexão, cada qual atuando como um roteador ou como um *switch*. Dois tipos de comutação são tradicionalmente utilizados na rede: comutação de circuitos e comutação de pacotes. A camada de rede foi projetada como uma rede de comutação de pacotes.

A camada de rede supervisiona o processamento dos pacotes pelas redes físicas subjacentes. A entrega de um pacote pode ser direta ou indireta. Duas categorias de encaminhamento foram definidas: encaminhamento baseado no endereço de destino do datagrama IP e encaminhamento baseado no rótulo anexado a um datagrama IP.

O IPv4 é um protocolo não confiável e não orientado à conexão. É responsável pela entrega de pacotes da origem ao destino. Os pacotes na camada IP são denominados datagramas. Os identificadores usados na camada IP da pilha de protocolos TCP/IP são denominados endereços IP. Um endereço IPv4 apresenta 32 *bits* de comprimento e é dividido em duas partes: o prefixo e o sufixo. Todos os endereços em um bloco têm o mesmo prefixo, cada endereço tem um sufixo diferente.

O Protocolo de Mensagens de Controle da Internet (ICMP – Internet Control Message Protocol) dá suporte ao não confiável e não orientado a conexão Protocolo Internet (IP).

Para ser capaz de rotear um pacote, um roteador precisa de uma tabela de roteamento. Os protocolos de roteamento são programas da camada de aplicação específicos cujo dever é atualizar as tabelas de roteamento.

O *multicast* consiste no envio da mesma mensagem para mais de um destinatário ao mesmo tempo. O Protocolo de Gerenciamento de Grupos Internet (IGMP – Internet Group Management Protocol) é um protocolo envolvido na coleta de informações locais sobre participação em grupos.

● IPv6, a versão mais atual do Protocolo Internet, tem um espaço de endereçamento de 128 *bits*. ● IPv6 usa a notação hexadecimal com dois pontos e disponibiliza alguns métodos de abreviação.

4.7 ATIVIDADES PRÁTICAS

4.7.1 Testes

Diversos testes interativos relativos a este capítulo podem ser encontrados no *site* www.mhhe.com/forouzan. Recomendamos que o estudante realize os testes para verificar seu entendimento do material apresentado antes de continuar com as atividades práticas.

Questões

4-1 Por que o protocolo da camada de rede precisa fornecer serviços de empacotamento para a camada de transporte? Por que a camada de transporte não é capaz de enviar os segmentos sem antes encapsulá-los em datagramas?

4-2 Por que o roteamento é de responsabilidade da camada de rede? Em outras palavras, por que o roteamento não pode ser feito na camada de transporte ou na camada de enlace de dados?

4-3 Quais são as diferenças entre o processo de roteamento de um pacote da origem até o destino e o processo de encaminhamento de um pacote em cada roteador?

4-4 Qual informação de um pacote é utilizada para tomar a decisão de encaminhamento em cada uma das seguintes abordagens de comutação?

a. abordagem de datagramas

b. abordagem de circuitos virtuais

4-5 Se um rótulo em um serviço orientado a conexão tem 8 *bits*, quantos circuitos virtuais podem ser estabelecidos ao mesmo tempo?

4-6 Liste as três etapas envolvidas na comutação usando a abordagem de circuitos virtuais.

4-7 Algum dos seguintes serviços está presente na camada de rede da pilha de protocolos TCP/IP? Em caso negativo, explique o porquê.

a. controle de fluxo

b. controle de erros

c. controle de congestionamento

4-8 Liste quatro tipos de atrasos em uma rede de comutação de pacotes.

4-9 Na Figura 4.10, considere que a conexão entre R1 e R2 foi aumentada para 170 Kbps e que a conexão entre a estação de origem e R1 foi reduzida para 140 Kbps. Qual é a vazão entre a origem e o destino após essas mudanças? Qual é o enlace que tornou-se o gargalo nesse caso?

4-10 O valor do campo de comprimento do cabeçalho em um pacote IPv4 pode ser inferior a 5? Quando ele é exatamente 5?

4-11 Uma estação está enviando 100 datagramas para outra estação. Se o número de identificação do primeiro datagrama for 1.024, qual será o número de identificação do último?

4-12 Um fragmento IP chegou com um valor de deslocamento de 100. Quantos *bytes* de dados foram originalmente enviados pela fonte antes dos dados desse fragmento?

4-13 No endereçamento sem classes, considere que conheçamos o primeiro e o último endereços do bloco. É possível determinar o comprimento do prefixo? Se a resposta for sim, descreva o processo.

4-14 No endereçamento sem classes, considere que conheçamos o primeiro endereço do bloco e seu número de endereços. É possível determinar o comprimento do prefixo? Se a resposta for sim, descreva o processo.

4-15 No endereçamento sem classes, dois blocos diferentes podem ter o mesmo comprimento de prefixo? Explique.

4-16 No endereçamento sem classes, considere que conheçamos o primeiro endereço e um outro endereço do bloco (não necessariamente o último). É possível determinar o comprimento do prefixo? Explique.

4-17 Um ISP possui um bloco de 1.024 endereços. Ele precisa dividir os endereços entre 1.024 clientes. Deverá criar sub-redes? Explique sua resposta.

4-18 Liste os três protocolos auxiliares da camada de rede da pilha de protocolos TCP/IP que foram concebidos para auxiliar o protocolo IPv4.

4-19 Em um datagrama IPv4, o valor do campo de comprimento do cabeçalho (HLEN) é $(6)_{16}$. Quantos *bytes* de opções foram adicionados ao pacote?

4-20 O campo TTL em um datagrama pode assumir algum dos seguintes valores? Explique sua resposta.

a. 23 **b.** 0 **c.** 1 **d.** 301

4-21 Compare o campo de protocolo na camada de rede com os números de porta na camada de transporte. Qual é o propósito em comum desses campos? Por que precisamos de dois campos para os números de porta, mas somente um campo de protocolo? Por que o tamanho do campo de protocolo é apenas metade do tamanho de cada campo de número de porta?

4-22 Qual (ou quais) campo no datagrama é responsável por ligar todos os fragmentos pertencentes a um datagrama original?

4-23 Considere que um computador de destino receba vários pacotes de uma origem. Como ele pode ter certeza de que os fragmentos pertencentes a um datagrama não estão misturados com os fragmentos pertencentes a outro datagrama?

4-24 Explique por que usar o MPLS implica acrescentar uma nova camada à pilha de protocolos TCP/IP. Onde esta camada está localizada?

4-25 Explique por que a Internet não cria uma mensagem de relatório para reportar erros em um datagrama IP que carrega uma mensagem ICMP.

4-26 Quais são os endereços IP de origem e destino em um datagrama que carrega uma mensagem ICMP de relatório de erro enviada por um roteador?

4-27 Em um grafo, se sabemos que o caminho mais curto do nó A ao nó G é (A→B→E→G), qual é o caminho mais curto do nó G ao nó A?

4-28 Em um grafo, considere que o caminho mais curto do nó A ao nó H seja A→B→H. E também que o caminho mais curto do nó H ao nó N seja H→G→N. Qual é o caminho mais curto do nó A ao nó N?

4-29 Explique por que um roteador usando roteamento por estado de enlace precisa receber toda a LSDB antes de criar e usar sua tabela de roteamento. Em outras palavras, por que o roteador não pode criar sua tabela de roteamento com uma LSDB recebida parcialmente?

4-30 O algoritmo de roteamento por vetor de caminhos é mais semelhante ao algoritmo de roteamento por vetor de distâncias ou ao algoritmo de roteamento por estado de enlace? Explique.

4-31 Liste três tipos de sistemas autônomos (AS) descritos no texto e os compare entre si.

4-32 Explique o conceito de contagem de saltos no RIP. Você saberia explicar por que nenhum salto é contado entre N1 e R1 na Figura 4.70?

4-33 Considere que haja um AS isolado executando o RIP. Podemos dizer que temos pelo menos dois tipos diferentes de tráfego de datagramas nesse AS. O primeiro tipo carrega as mensagens trocadas entre as estações; já o segundo, as mensagens pertencentes ao RIP. Qual é a diferença entre os dois tipos de tráfego quando consideramos os endereços IP de origem e de destino? Isto mostra que os roteadores também precisam de endereços IP?

4-34 O roteador A envia duas mensagens RIP para dois roteadores que são seus vizinhos imediatos, B e C. Os dois datagramas que transportam as mensagens têm o mesmo endereço IP de origem? Os dois datagramas têm os mesmos endereços IP de destino?

4-35 A qualquer momento, uma mensagem RIP pode chegar em um roteador executando o RIP como protocolo de roteamento. Isto significa que o processo do RIP deve estar sendo executado o tempo todo?

4-36 Por que você acha que o RIP usa UDP em vez de TCP?

4-37 Dizemos que o OSPF é um protocolo intradomínio hierárquico, mas que o RIP não é. Qual é a razão por trás dessa afirmação?

4-38 Em um AS muito pequeno utilizando OSPF, é mais eficiente usar uma única área (*backbone*) ou várias áreas?

4-39 Por que você acha que é necessária apenas uma mensagem de atualização no RIP, porém várias mensagens de atualização no OSPF?

4-40 As mensagens OSPF são trocadas entre os roteadores. Isto significa que os processos OSPF precisam estar executando o tempo todo para que seja possível receber uma mensagem OSPF no momento em que ela chega?

4-41 As mensagens OSPF e ICMP são diretamente encapsuladas em um datagrama IP. Se interceptarmos um datagrama IP, como podemos determinar se a sua carga útil pertence ao OSPF ou ao ICMP?

4-42 Explique qual é o tipo de estado de enlace OSPF anunciado em cada um dos seguintes casos:

a. Um roteador precisa anunciar a existência de outro roteador no final de um enlace ponto a ponto.

b. Um roteador precisa anunciar a existência de duas redes *stub* em uma rede de transição.

c. Um roteador designado anuncia uma rede como um nó.

4-43 Um roteador pode combinar o anúncio de um enlace e de uma rede em uma única mensagem de atualização de estado de enlace?

4-44 Explique por que podemos ter protocolos de roteamento intradomínio diferentes em AS diferentes, mas precisamos ter apenas um protocolo de roteamento interdomínios em toda a Internet.

4-45 Por que você acha que o BGP utiliza os serviços do TCP em vez dos serviços do UDP?

4-46 Explique por que a política de roteamento pode ser implementada em um roteamento interdomínios, mas não pode ser implementada em um roteamento intradomínio.

4-47 Explique quando cada um dos seguintes atributos pode ser usado no BGP:

a. LOCAL-PREF

b. AS-PATH

c. NEXT-HOP

4-48 Compare uma comunicação *multicast* com múltiplas comunicações *unicast*.

4-49 Quando enviamos um e-mail a múltiplos destinatários, estamos utilizando comunicação *multicast* ou múltiplas comunicações *unicast*? Explique sua resposta.

4-50 Quais dos seguintes endereços são endereços *multicast*?

a. 224.8.70.14

b. 226.17.3.53

c. 240.3.6.25

4-51 Qual é o grupo de cada um dos seguintes endereços *multicast* (bloco de controle da rede local, bloco de controle inter-redes, bloco SSM, bloco GLOP ou bloco de alcance administrativo)?

a. 224.0.1.7

b. 232.7.14.8

c. 239.14.10.12

4-52 Uma estação pode ter mais do que um endereço *multicast*? Explique.

4-53 Um roteador *multicast* está conectado a quatro redes. Caso o interesse de cada rede seja o mostrado a seguir, qual é a lista de grupos que deve ser anunciada pelo roteador?

a. N1: {G1, G2, G3}

b. N2: {G1, G3}

c. N3: {G1, G4}

d. N4: {G1, G3}

4-54 É óbvio que precisamos ter árvores de abrangência (*spanning trees*) tanto na comunicação *unicast* como na *multicast*. Quantas folhas da árvore estão envolvidas na transmissão em cada caso?

a. transmissão *unicast*

b. transmissão *multicast*

4-55 Considere que haja 20 *hosts* em um pequeno AS e que existam apenas quatro grupos nele. Encontre o número de árvores de abrangência (*spanning trees*) em cada uma das seguintes abordagens:

a. árvore baseada na origem

b. árvore compartilhada pelo grupo

4-56 Dizemos que um roteador DVMRP cria uma árvore de menor custo sob demanda. Qual é o significado dessa afirmação? Qual é a vantagem de criar árvores de menor custo somente sob demanda?

4-57 Liste três passos que um roteador DVMRP usa para criar uma árvore baseada na origem. Qual é o passo responsável por criar a parte da árvore que vai da origem até o roteador atual? Qual é o passo responsável por criar uma árvore de *broadcast* tendo o roteador como raiz? Qual é o passo responsável por transformar a árvore de *broadcast* em uma árvore de *multicast*?

4-58 Por que um roteador MOSPF consegue criar o caminho mais curto tendo a origem como a raiz em um único passo, enquanto o DVMRP precisa de três passos para fazê-lo?

4-59 Explique por que o PIM é denominado *Multicast Independente de Protocolo*.

4-60 Qual é o modo de operação do PIM que utiliza o primeiro e o terceiro passos do DVMRP? Quais são esses dois passos?

4-61 Por que o *broadcast* da primeira ou de algumas das primeiras mensagens não é um problema no PIM-DM, mas o é no PIM-SM?

4-62 Explique as vantagens do IPv6 em relação ao IPv4.

4-63 Explique o uso do campo de fluxo no IPv6. Qual é a potencial aplicação deste campo?

4-64 Cite as diferenças entre endereços compatíveis e mapeados e explique suas aplicações.

4-65 Lista três protocolos da camada de rede IPv4 que são combinados em um único protocolo no IPv6.

4-66 Qual é a razão para se incluir o cabeçalho IP e os primeiros 8 *bytes* de dados do datagrama nas mensagens ICMP de relatório de erro?

Problemas

4-1 Em um datagrama IPv4, o valor do campo de comprimento total é $(00A0)_{16}$ e o valor do comprimento do cabeçalho (HLEN) é $(5)_{16}$. Quantos *bytes* de carga útil estão sendo transportados pelo datagrama? Qual é a eficiência (razão entre o comprimento da carga útil e o comprimento total) de tal datagrama?

4-2 Um datagrama IP chegou com a seguinte informação parcial no cabeçalho (em hexadecimal):

```
45000054 00030000 2006...
```

a. Qual é o tamanho do cabeçalho?
b. Há alguma opção no pacote?
c. Qual é o tamanho dos dados?
d. O pacote está fragmentado?
e. Quantos roteadores mais o pacote pode visitar?
f. Qual é o número do protocolo referente à carga útil sendo transportada pelo pacote?

4-3 Quais campos do cabeçalho principal do IPv4 podem mudar de um roteador para o outro?

4-4 Determine se um datagrama com a seguinte informação é um primeiro fragmento, um fragmento intermediário, o último fragmento ou o único fragmento (não há fragmentação):

a. O *bit M* vale 1 e o campo de deslocamento vale zero.
b. O *bit M* vale 1 e o valor do campo de deslocamento é diferente de zero.

4-5 Descreva brevemente como podemos nos proteger dos seguintes ataques contra a segurança de um sistema:

a. escuta de pacotes
b. modificação de pacotes
c. falsificação de IP

4-6 Qual é o tamanho do espaço de endereços em cada um dos seguintes sistemas?

a. Um sistema em que cada endereço tem apenas 16 *bits*.
b. Um sistema no qual cada endereço é composto por seis dígitos hexadecimais.
c. Um sistema no qual cada endereço é composto por quatro dígitos octais.

4-7 Reescreva os seguintes endereços IP usando a notação binária:

a. 110.11.5.88
b. 12.74.16.18
c. 201.24.44.32

4-8 Reescreva os seguintes endereços IP utilizando notação decimal com pontos:

a. 01011110 10110000 01110101 00010101
b. 10001001 10001110 11010000 00110001
c. 01010111 10000100 00110111 00001111

4-9 Determine a classe dos seguintes endereços IP no endereçamento com classes:

a. 130.34.54.12
b. 200.34.2.1
c. 245.34.2.8

4-10 Determine a classe dos seguintes endereços IP no endereçamento com classes:

a. 01110111 11110011 10000111 11011101
b. 11101111 11000000 11110000 00011101
c. 11011111 10110000 00011111 01011101

4-11 No endereçamento sem classes, mostre o espaço de endereços completo como um único bloco usando a notação CIDR.

4-12 No endereçamento sem classes, qual é o tamanho do bloco (N) quando o comprimento do prefixo (n) assume cada um dos seguintes valores?

a. $n = 0$
b. $n = 14$
c. $n = 32$

4-13 No endereçamento sem classes, qual é o comprimento do prefixo (n) quando o tamanho do bloco (N) assume cada um dos seguintes valores?

a. $N = 1$ **b.** $N = 1024$ **c.** $N = 232$

4-14 Transforme cada um dos comprimentos de prefixo a seguir em uma máscara em notação decimal com pontos:

a. $n = 0$ **b.** $n = 14$ **c.** $n = 30$

4-15 Transforme cada uma das máscaras a seguir em um comprimento de prefixo:

a. 255.224.0.0
b. 255.240.0.0
c. 255.255.255.128

4-16 Qual das seguintes opções não pode corresponder a uma máscara no CIDR?

a. 255.225.0.0
b. 255.192.0.0
c. 255.255.255.6

4-17 Cada um dos seguintes endereços pertence a um bloco. Determine o primeiro e o último endereços em cada bloco.

a. 14.12.72.8/24
b. 200.107.16.17/18
c. 70.110.19.17/16

4-18 Mostre os *n bits* mais à esquerda dos seguintes endereços de rede/máscara que podem ser usados em uma tabela de roteamento (ver Figura 4.45).

a. 170.40.11.0/24
b. 110.40.240.0/22
c. 70.14.0.0./18

4-19 Explique como o DHCP pode ser usado quando o tamanho do bloco atribuído a uma organização é menor do que o número de *hosts* na organização.

4-20 Compare o NAT e o DHCP. Ambos podem amenizar o problema da falta de endereços em uma organização, mas usando estratégias diferentes.

4-21 Considere uma internet com um espaço de endereços de 8 *bits*. Os endereços são divididos igualmente entre quatro redes (N0 a N3). A comunicação inter-redes é feita por meio de um roteador com quatro interfaces (m0 a m3). Mostre a estrutura da rede e a tabela de roteamento (com duas colunas: prefixo em binário e número da interface) para o único roteador conectando as redes. Atribua um endereço de rede a cada rede.

4-22 Considere uma internet com um espaço de endereços de 12 *bits*. Os endereços são divididos igualmente entre oito redes (N0 a N7). A comunicação inter-redes é feita por meio de um roteador com oito interfaces (m0 a m7). Mostre a estrutura da rede e a tabela de roteamento (com duas colunas: prefixo em binário e número da interface) para o único roteador conectando as redes. Atribua um endereço de rede a cada rede.

4-23 Considere uma internet com um espaço de endereços de 9 *bits*. Os endereços são divididos entre três redes (N0 a N2), as quais recebem 64, 192 e 256 endereços, respectivamente. A comunicação inter-redes é feita por meio de um roteador com três interfaces (m0 a m2). Mostre a estrutura da rede e a tabela de roteamento (com duas colunas: prefixo em binário e número da interface) para o único roteador conectando as redes. Atribua um endereço de rede a cada rede.

4-24 Combine os três blocos de endereços seguintes em um único bloco:

a. 16.27.24.0/ **26**
b. 16.27.24.64/ **26**
c. 16.27.24.128/ **25**

4-25 Uma empresa de grande porte com um grande bloco de endereços (12.44.184.0/**21**) é dividida em uma empresa de médio porte usando o bloco de endereços (12.44.184.0/**22**) e duas empresas de pequeno porte. Se a primeira empresa de pequeno porte utiliza o bloco (12.44.188.0/**23**), qual é o bloco restante que pode ser utilizado pela segunda empresa de pequeno porte? Explique como os datagramas destinados às duas empresas de pequeno porte podem ser roteados corretamente a essas empresas apesar de seus blocos de endereços ainda serem parte da empresa original.

4-26 Um ISP recebe o bloco de endereços 16.12.64.0/**20**. O ISP precisa alocar endereços para oito organizações, cada uma com 256 endereços.

a. Determine o número e a faixa de endereços no bloco do ISP.
b. Determine a faixa de endereços alocados a cada organização e a faixa de endereços não alocados.
c. Desenhe um diagrama com a distribuição de endereços e a tabela de roteamento.

4-27 Um ISP recebe o bloco de endereços 80.70.56.0/**21.** O ISP precisa alocar endereços para duas organizações tendo cada uma

500 endereços, para duas organizações tendo cada uma 250 endereços e para três organizações tendo cada uma 50 endereços.

a. Determine o número e a faixa de endereços no bloco do ISP.

b. Determine a faixa de endereços alocados a cada organização e a faixa de endereços não alocados.

c. Desenhe um diagrama com a distribuição de endereços e a tabela de roteamento.

4-28 Uma organização recebe o bloco de endereços **130.56.0.0/16.** Nesse contexto, o administrador da organização deseja criar 1.024 sub-redes.

a. Determine o número de endereços em cada sub-rede.

b. Determine o prefixo das sub-redes.

c. Determine o primeiro e o último endereços da primeira sub-rede.

d. Determine o primeiro e o último endereços da última sub-rede.

4-29 O roteador R1 da Figura 4.48 pode receber um pacote com endereço de destino 140.24.7.194? O que acontecerá com o pacote se isso ocorrer?

4-30 Considere que o roteador R2 da Figura 4.48 receba um pacote cujo endereço de destino seja 140.24.7.42. Como este pacote é encaminhado ao seu destino final?

4-31 Considere que a menor distância dos nós ***a***, ***b***, ***c*** e ***d*** até o nó ***y*** e que os custos do nó ***x*** até os nós ***a***, ***b***, ***c*** e ***d*** sejam dados a seguir:

$\mathbf{D}_{ay} = 5$ $\mathbf{D}_{by} = 6$ $\mathbf{D}_{cy} = 4$ $\mathbf{D}_{dy} = 3$

$\mathbf{c}_{xa} = 2$ $\mathbf{c}_{xb} = 1$ $\mathbf{c}_{xc} = 3$ $\mathbf{c}_{xd} = 1$

Qual é a menor distância entre o nó ***x*** e o nó ***y***, $\mathbf{D}_{xy}$, de *acordo* com a equação de Bellman-Ford?

4-32 Considere que um roteador usando RIP tenha 10 entradas em sua tabela de roteamento no instante t_1. Seis dessas entradas ainda são válidas no instante t_2. Quatro delas expiraram 70, 90, 110 e 210 segundos antes do instante t_2. Determine o número de temporizadores periódicos, temporizadores de expiração e temporizadores de coleta de lixo (*garbage collection*) em execução nos instantes t_1 e t_2.

4-33 Quando um roteador executando OSPF envia cada uma das seguintes mensagens?

a. olá (*hello*)

b. descrição da base de dados (*database description*)

c. solicitação de estado de enlace (*link-state request*)

d. atualização de estado de enlace (*link-state update*)

e. confirmação de estado de enlace (*link-state acknowledgment*)

4-34 Para entender o funcionamento do algoritmo de vetor de distâncias da Tabela 4.4, vamos aplicá-lo a uma internet contendo quatro nós, mostrada na Figura 4.111.

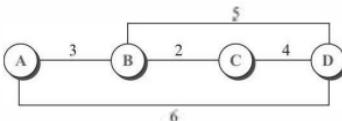
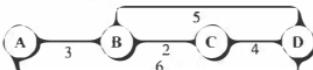



Figura 4.111 Esquema para o Problema 4-34.

Considere que todos os nós são primeiramente inicializados, e também que o algoritmo seja aplicado a cada nó na sequência (A, B, C, D), um de cada vez. Mostre que o processo converge e que todos os nós obterão vetores de distâncias estáveis.

4-35 No roteamento por vetor de distâncias, boas notícias (redução do custo em um enlace) se propagam rapidamente. Em outras palavras, se a distância em um conexão física diminui, todos os nós são rapidamente informados sobre o fato e atualizam seus vetores. Na Figura 4.112, consideramos que uma internet que contém quatro nós esteja estável, mas de repente a distância entre os nós A e D, que atualmente vale 6, é reduzida para 1 (provavelmente devido a alguma melhoria na qualidade da conexão). Mostre como essa boa notícia é propagada e determine o novo vetor de distâncias para cada nó após sua estabilização.

Figura 4.112 Esquema para o Problema 4-35.

4-36 No roteamento por vetor de distâncias, más notícias (aumento no custo de um enlace) se propagam lentamente. Em outras palavras, se a distância em uma conexão física aumenta, às vezes leva muito tempo para todos os nós serem informados da má notícia. Na Figura 4.112 (ver problema anterior), considere que a internet composta por quatro nós esteja estável, mas de repente a distância entre os nós B e C, que era de 2, passe a valer infinito (há uma falha na conexão). Mostre como essa má notícia é propagada e determine o novo vetor de distâncias para cada novo nó após a estabilização deles. Considere que a aplicação usa um temporizador periódico para acionar atualizações para os vizinhos (as atualizações não são mais acionadas quando há uma mudança), e assuma também que, se um nó receber um custo mais elevado proveniente do mesmo vizinho que o informou anteriormente, ele usa o novo custo, pois isto significa que a informação antiga não é mais válida. Para acelerar a estabilização, a implementação também suspende uma rota quando o próximo salto não estiver acessível.

4-37 Na área da ciência da computação, quando nos deparamos com um algoritmo, muitas vezes precisamos descobrir sua complexidade (quantas operações precisamos realizar). Para descobrir a complexidade do algoritmo de vetor de distâncias, determine o número de operações que um nó precisa realizar quando recebe um vetor de um vizinho.

4-38 Considere que a rede da Figura 4.113 utilize roteamento por vetor de distâncias com as tabelas de roteamento mostradas para cada nó.

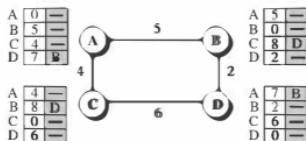



Figura 4.113 Esquema para o Problema 4-38.

Se cada nó anuncia periodicamente seus vetores aos vizinhos usando a estratégia de envenenamento reverso (*poisoned reverse*), qual será o vetor de distâncias anunciado no instante apropriado nos seguintes casos?

a. de A para B?

b. de C para D?

c. de D para B?

d. de C para A?

4-39 Considere que a rede da Figura 4.113 (problema anterior) utilize roteamento por vetor de distâncias com as tabelas de roteamento mostradas para cada nó. Se cada nó anuncia periodicamente seus vetores aos seus vizinhos usando a estratégia de horizonte dividido (*split horizon*), qual será o vetor de distâncias anunciado no instante apropriado nos seguintes casos?

a. de A para B?

b. de C para D?

c. de D para B?

d. de C para A?

4-40 Considere que a rede da Figura 4.113 (Problema 4-38) utilize roteamento por vetor de distâncias com as tabelas de roteamento mostradas para cada nó. Se o nó E for adicionado à rede com um custo de enlace de 1 até o nó D, determine quais serão as novas tabelas de roteamento de cada nó sem usar o algoritmo de vetor de distâncias.

4-41 Crie a tabela de roteamento para o nó A da Figura 4.65.

4-42 Crie a árvore de menor custo e a tabela de roteamento para o nó G da Figura 4.63.

4-43 Crie a árvore de menor custo e a tabela de roteamento para o nó B da Figura 4.63.

4-44 Use o algoritmo de Dijkstra (Tabela 4.4) para determinar a árvore de menor custo e a tabela de roteamento para o nó A da Figura 4.114.

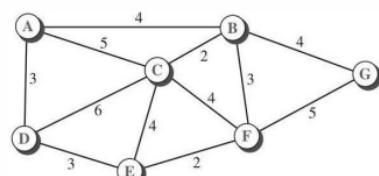



Figura 4.114 Esquema para o Problema 4-44.

4-45 Na área da ciência da computação, quando nos deparamos com um algoritmo, muitas vezes precisamos descobrir sua complexidade (quantas operações precisamos realizar). Para descobrir a complexidade do algoritmo de Dijkstra, determine o número de operações de busca que precisamos realizar para

determinar o caminho mais curto para um único nó quando o número de nós é *n*.

4-46 Considere que A, B, C, D e E na Figura 4.115 sejam sistemas autônomos (AS). Determine o vetor de caminhos para cada AS usando o algoritmo da Tabela 4.6. Considere que o melhor caminho nesse caso seja o caminho passando pelo menor número de AS, e também que o algoritmo primeiro inicializa cada AS e é então aplicado a cada nó na sequência (A, B, C, D, E), uma de cada vez. Mostre que o processo converge e que todos os AS obterão vetores de caminhos estáveis.

4-47 Na Figura 4.79, considere que o protocolo de roteamento intra-AS usado por AS1 seja o OSPF, mas que o protocolo usado por AS2 seja o RIP. Explique como R5 pode determinar uma forma de rotear pacotes para N4.

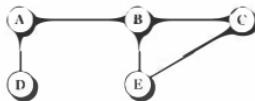



Figura 4.115 Esquema para o Problema 4-46.

4-48 Na Figura 4,79, considere que o protocolo de roteamento intra-AS usado por AS4 e AS3 seja o RIP. Explique como R8 pode determinar uma forma de rotear pacotes para N13.

4-49 Na Figura 4.116, considere que o roteador R3 recebeu três cópias do mesmo pacote provenientes da origem O, através de três interfaces, m0, m1 e m2. Qual é o pacote que deve ser encaminhado pela interface m3 para o restante da rede se o roteador R3 usar a estratégia RPF?

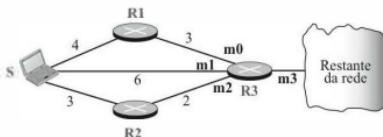



Figura 4.116 Esquema para o Problema 4-49.

4-50 Considere que *m* seja muito menor do que *n* e que o roteador R esteja ligado a *n* redes, mas apenas *m* destas redes estejam interessadas em receber pacotes relativos ao grupo G. Como o roteador R pode enviar uma cópia do pacote apenas para as redes interessadas no grupo G?

4-51 Na rede da Figura 4.117, determine as árvores de menor custo para o roteador R em dois casos: primeiro, se a rede estiver usando OSPF; segundo, se a rede estiver usando MOSPF com a origem conectada ao roteador marcado como O. Considere que todos os roteadores tenham interesse no grupo *multicast* correspondente.

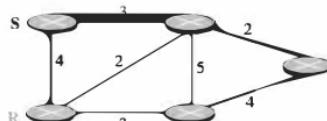



Figura 4.117 Esquema para o Problema 4-51.

4-52 Qual é o tamanho de uma mensagem RIP que anuncia uma única rede? Qual é o tamanho de uma mensagem RIP que anuncia *n* redes?

4-53 Um roteador usando RIP apresenta a seguinte tabela de roteamento. Mostre a mensagem de atualização RIP enviada por esse roteador.

Destino	Custo	Próximo roteador
Rede1	4	B
Rede2	2	C
Rede3	1	F
Rede4	5	G

4-54 Compare o cabeçalho IPv4 com o cabeçalho IPv6. Crie uma tabela para comparar cada campo.

4-55 Escreva a notação hexadecimal com dois pontos não abreviada para os seguintes endereços IPv6:

a. Um endereço com 64 0s seguidos de 32 pares de *bits* (01)s.

b. Um endereço com 64 0s seguidos de 32 pares de *bits* (10)s.

c. Um endereço com 64 pares de *bits* (01)s.

d. Um endereço com 32 quadras de *bits* (0111)s.

4-56 Escreva os seguintes endereços em notação abreviada:

a. 0000:FFFF:FFF:0000:0000:0000:0000:0000
b. 1234:2346:3456:0000:0000:0000:0000:FFFF
c. 0000:0001:0000:0000:0000:FFFF:1200:1000
d. 0000:0000:0000:0000:FFFF:FFFF:24.123.12.6

4-57 Descompacte os seguintes endereços e mostre o endereço IPv6 completo (sem abreviações):

a. ::2222
b. 1111::
c. B:A:CC::1234:A

4-58 Mostre a forma original (sem abreviações) dos seguintes endereços IPv6:

a. ::2
b. 0:23::0
c. 0:A::3

4-59 Com base na Tabela 4.7, qual é o bloco ou sub-bloco associado a cada um dos seguintes endereços IPv6:

a. FE80::12/ **10**
b. FD23::/ **7**
c. 32::/ **3**

4-60 O bloco 2000:1234:1423/48 é atribuído a uma organização. Qual é o CIDR para os blocos na primeira e na segunda sub-redes dessa organização?

4.8 EXPERIMENTOS DE SIMULAÇÃO

4.8.1 *Applets*

Criamos *applets* Java para demonstrar alguns dos principais conceitos discutidos neste capítulo. Recomendamos que o estudante ative esses *applets* no *site* www.mhhe.com/forouzan o e examine cuidadosamente os protocolos em ação.

4.8.2 Experimentos de laboratório

Neste capítulo, usamos o Wireshark para capturar e analisar alguns pacotes trocados na camada de rede. Usamos Wireshark e alguns outros utilitários de administração de rede de computadores. Consulte o *site* do livro para obter detalhes completos dos experimentos de laboratório.

Lab4-1 No primeiro experimento de laboratório, analisamos o protocolo IP, capturando e estudando datagramas IP.

Lab4-2 No segundo experimento de laboratório, capturamos e estudamos os pacotes ICMP gerados por outros programas utilitários, tais como os programas *ping* e *traceroute*.

4.9 TAREFAS DE PROGRAMAÇÃO

Escreva o código-fonte, compile e teste os seguintes programas usando a linguagem de programação de sua preferência:

Prg4-1 Escreva um programa para simular o algoritmo de vetor de distâncias (Tabela 4.4).

Prg4-2 Escreva um programa para simular o algoritmo de estado de enlace (Tabela 4.5).

Prg4-3 Escreva um programa para simular o algoritmo de vetor de caminhos (Tabela 4.6).

CAPÍTULO

5 CAMADA DE ENLACE DE DADOS: REDES COM FIOS

Nos Capítulos 2 a 4, discutimos as camadas de aplicação, transporte e rede. Neste capítulo e no seguinte, discutiremos a camada de enlace de dados. A pilha de protocolos TCP/IP não especifica qualquer protocolo nas camadas de enlace de dados e físicas; estas são os territórios das redes que, quando conectadas, formam a Internet. Conforme discutimos no Capítulo 1, essas redes, com ou sem fios, fornecem serviços para as três camadas superiores da pilha de protocolos TCP/IP. É um primeiro indício de que existem vários protocolos padronizados no mercado atualmente. Por isso, discutimos a camada de enlace de dados em dois capítulos: neste, discutimos os conceitos gerais envolvidos dando foco às redes com fios, também conhecidas como redes cabeadas ou guiadas; no próximo, discutiremos as redes sem fios. Este capítulo está dividido em sete seções:

- Na primeira seção, introduzimos o conceito de nós e enlaces, discutimos os tipos de enlaces e mostramos como a camada de enlace de dados é, na realidade, dividida em duas subcamadas: Controle de Enlace de Dados (DLC – Data Link Control) e Controle de Acesso ao Meio (MAC – Media Access Control).
- Na segunda seção, discutimos a subcamada de controle de enlace de dados (DLC) da camada de enlace de dados e explicamos os serviços fornecidos pela subcamada, como enquadramento, controle de fluxo e de erros, e detecção de erros.
- Na terceira seção, discutimos a subcamada de controle de acesso ao meio (MAC) da camada de enlace de dados. Explicamos diferentes abordagens usadas por ela, como acesso aleatório, acesso controlado e canalização.
- Na quarta seção, discutimos o endereçamento na camada de enlace e como o endereço da camada de enlace de um nó pode ser encontrado usando o Protocolo de Resolução de Endereços (ARP – Address Resolution Protocol).
- Na quinta seção, apresentamos as LANs com fios e em particular a Ethernet, o protocolo de LAN dominante atualmente. Navegamos pelas diferentes gerações da Ethernet e mostramos como ela evoluiu.
- Na sexta seção, discutimos outras redes com fios encontradas na Internet atual, como as redes ponto a ponto e comutadas.
- Na sétima seção, discutimos os dispositivos de conexão utilizados nas três camadas mais baixas da pilha de protocolos TCP/IP, como *hubs*, *switches* da camada de enlace e roteadores.

5.1 INTRODUÇÃO

No Capítulo 4, aprendemos que a comunicação na camada de rede é *host* a *host*. Embora possa ser fragmentado e remontado, um datagrama é uma unidade de dados enviada de um *host*, localizado

em algum lugar do mundo, para outro *host*, em algum outro lugar no mundo. A Internet, porém, é uma combinação de redes ligadas entre si por meio de dispositivos de conexão (roteadores ou *switches*). Se um datagrama precisar viajar de uma estação para outra, precisará passar por tais redes.

A Figura 5.1 mostra a comunicação entre Alice e Bob, usando o mesmo cenário que usamos nos últimos três capítulos. A comunicação na camada de enlace de dados, entretanto, é composta por cinco conexões lógicas distintas entre as camadas de enlace de dados no caminho de uma estação à outra.

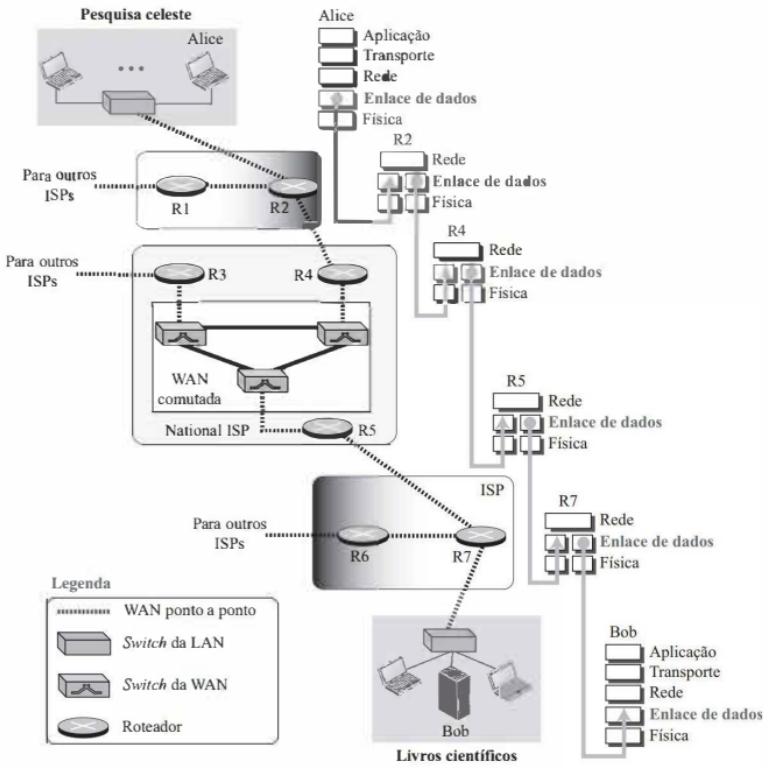



Figura 5.1 Comunicação na camada de enlace de dados.

A camada de enlace de dados do computador de Alice se comunica com a camada de enlace de dados do roteador R2. A camada de enlace de dados do roteador R2 se comunica com a camada de enlace de dados do roteador R4, e assim por diante. Finalmente, a camada de enlace de dados do roteador R7 se comunica com a camada de enlace de dados do computador de Bob. Apenas uma camada de enlace de dados atua tanto na origem como no destino, mas duas camadas de enlace de

dados atuam em cada roteador. A razão é que os computadores de Alice e Bob estão conectados a uma única rede; cada roteador, porém, recebe os dados de entrada vindos de uma rede e envia os dados de saída para outra rede.

5.1.1 Nós e enlaces

Embora a comunicação nas camadas de aplicação, transporte e rede sejam fim a fim, a comunicação na camada de enlace de dados é nó a nó, ou salto a salto. Como vimos nos capítulos anteriores, uma unidade de dados de um ponto na Internet precisa passar por muitas redes (LANs e WANs) para chegar a outro ponto. Essas LANs e WANs são conectadas por roteadores. É comum denominar as duas estações finais e os roteadores como **nós** e as redes entre eles como **enlaces**. Na Figura 5.2, mostramos uma representação simples de enlaces e nós quando o caminho da unidade de dados consiste em apenas seis nós.

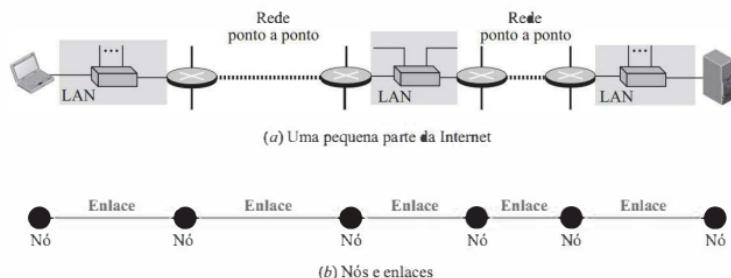



Figura 5.2 Nós e enlaces.

O primeiro nó é a estação de origem; o último é a estação de destino. Os outros quatro nós são quatro roteadores. O primeiro, o terceiro e o quinto enlaces representam três LANs; o segundo e o quarto enlaces representam as duas WANs.

5.1.2 Dois tipos de enlaces

Apesar de dois nós estarem fisicamente conectados por um meio de transmissão como um cabo ou o ar, precisamos lembrar que a camada de enlace de dados controla a forma como o meio é usado. Podemos ter uma camada de enlace de dados que utiliza toda a capacidade do meio; e também uma que utiliza apenas uma parte da capacidade do enlace. Em outras palavras, podemos ter um *enlace ponto a ponto* ou um *enlace de broadcast*. Em um enlace ponto a ponto, o enlace é dedicado aos dois dispositivos; já em um enlace de *broadcast*, o enlace é compartilhado entre vários pares de dispositivos. Por exemplo, quando dois amigos usam os tradicionais telefones fixos para conversar, eles estão usando um enlace ponto a ponto; quando usam seus celulares, eles estão usando um enlace de *broadcast* (o ar é compartilhado por diversos usuários de telefones celulares).

5.1.3 Duas subcamadas

Para entender melhor a funcionalidade da camada de enlace e os serviços prestados por ela, podemos dividir a camada de enlace de dados em duas subcamadas: *Controle de Dados de Enlace*

(DLC – Data Link Control) e *Controle de Acesso ao Meio* (MAC – Media Access Control). Isto não é incomum porque, como veremos neste e no próximo capítulo, atualmente os protocolos de LAN usam a mesma estratégia. A subcamada de controle de enlace de dados lida com todas as questões comuns tanto aos enlaces ponto a ponto como aos de *broadcast*; a subcamada de controle de acesso ao meio lida apenas com questões específicas dos enlaces de *broadcast*. Ou seja, separamos esses dois tipos de enlaces da camada de enlace de dados, conforme mostra a Figura 5.3.

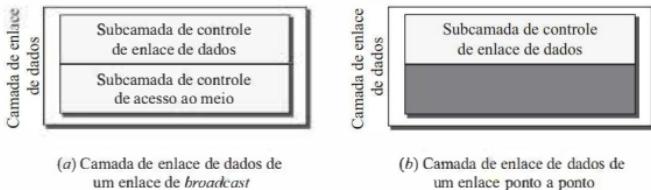



Figura 5.3 Dividindo a camada de enlace de dados em duas subcamadas.

Neste capítulo, começaremos discutindo a subcamada de controle de enlace de dados, que é comum a ambos os tipos de enlaces. Em seguida, discutiremos a subcamada de controle de acesso ao meio, usada somente nos enlaces de *broadcast*. Após a discussão sobre essas duas subcamadas, discutiremos um protocolo pertencente a cada categoria.

5.2 CONTROLE DE ENLACE DE DADOS

O controle de enlace de dados lida com procedimentos para a comunicação entre dois nós adjacentes – comunicação nó a nó – não importando se o enlace é dedicado ou de *broadcast*. As funções do **Controle de Enlace de Dados** (DLC – Link Data Control) incluem *enquadramento*, *controle de fluxo e de erro*, e detecção e correção de erros. Nesta seção, primeiramente discutimos o enquadramento, ou como organizar os *bits* que são transportados pela camada física. Em seguida, o controle de fluxo e de erro. Técnicas para a detecção de erros são abordadas no fim desta seção.

5.2.1 Enquadramento

A transmissão de dados na camada física consiste em mover *bits* na forma de um sinal, indo da origem até o destino. A camada física fornece mecanismos de sincronização de *bits* para garantir que o emissor e o receptor usem as mesmas durações e temporização de *bits*. Discutiremos a camada física no Capítulo 7.

A camada de enlace de dados, por outro lado, precisa agrupar os *bits* em quadros, de modo que cada quadro seja distinguível um do outro. Nosso sistema postal aplica um tipo de enquadramento. O simples ato de inserir uma carta em um envelope separa uma parte da informação da outra; o envelope serve como delimitador. Além disso, cada envelope define os endereços do remetente e do destinatário, algo necessário porque o transporte realizado pelo sistema postal é do tipo muitos para muitos.

O *enquadramento* na camada de enlace de dados define que uma mensagem vai de uma origem para um destino por meio da adição de um endereço de remetente e de um de destinatário. Este determina para onde o pacote deve ir; o endereço do remetente ajuda o destinatário a confirmar a recepção dos dados.

Embora a mensagem completa possa ser acondicionada em um quadro, isto normalmente não é feito. Uma razão é que tal quadro poderia ficar muito grande, tornando os mecanismos de controle de fluxo e de erros muito ineficientes. Quando uma mensagem é transportada em um quadro muito grande, mesmo o erro em um único *bit* exigiria a retransmissão do quadro todo. Quando uma mensagem é dividida em quadros menores, um erro em um único *bit* afeta apenas o quadro pequeno.

Comprimento dos quadros

Os quadros podem ter um tamanho fixo ou variável. No *enquadramento de tamanho fixo*, não é necessário definir os limites dos quadros: o próprio tamanho pode ser usado como um delimitador. Um exemplo de rede que adota esse tipo de enquadramento são as WANs ATM, que utilizam quadros de tamanho fixo denominados *células*. Discutiremos o ATM no final deste capítulo.

O cerne da nossa discussão neste capítulo refere-se ao *enquadramento de tamanho variável*, que é, de modo geral, usado em LANs. Nele, precisamos de algum mecanismo para definir o final de um quadro e o início do próximo. Historicamente, duas abordagens foram criadas com esse propósito: uma abordagem orientada a caracteres e outra orientada a *bits*.

Enquadramento orientado a caracteres

No *enquadramento orientado a caracteres* (ou *orientado a bytes*), os dados transportados são caracteres de 8 *bits* provenientes de algum sistema de codificação, tal como ASCII (ver o Apêndice A no *site* do livro). O cabeçalho, que normalmente transporta os endereços de origem e de destino, bem como outras informações de controle, como o rodapé (*trailer*), que transporta *bits* redundantes usados para detecção de erros, também são múltiplos de 8 *bits*. Para separar um quadro do outro, um **marcador** de 8 *bits* (1 *byte*) é adicionado ao início e ao fim de cada quadro. O marcador, composto por caracteres especiais dependentes do protocolo, sinaliza o início e o fim de um quadro. A Figura 5.4 mostra o formato de um quadro em um protocolo orientado a caracteres.

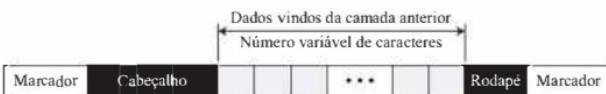



Figura 5.4 Um quadro em um protocolo orientado a caracteres.

O enquadramento orientado a caracteres era bastante popular quando as camadas de enlace de dados trocavam somente texto entre elas. O marcador podia ser qualquer caractere que não fosse usado na comunicação usando texto. Atualmente, entretanto, enviamos outros tipos de informação, como gráficos, áudio e vídeo, de modo que qualquer sequência usada como marcador pode também ser parte da informação. Se isto acontecer, o receptor, ao se deparar com essa sequência no meio dos dados, pensa que atingiu o fim do quadro. Para corrigir esse problema, uma estratégia de **preenchimento de *byte*** foi adicionada ao enquadramento orientado a caracteres. Na técnica de preenchimento de *byte* (ou preenchimento de caracteres), um *byte* especial é adicionado à seção de dados do quadro sempre que ele apresentar um caractere com a mesma sequência de *bits* que o marcador. A seção de dados é preenchida com um *byte* adicional, normalmente denominado *caractere de escape* (ESC) e consiste em uma sequência de *bits* predefinida. Sempre que o receptor encontra o caractere ESC, ele o remove da seção de dados e trata o próximo caractere como dados e não como um marcador de fim de quadro.

O preenchimento de *byte* por meio de um caractere de escape permite a presença de marcadores na seção de dados do quadro, mas cria outro problema. O que acontece se o texto contiver um ou mais caracteres de escape seguidos por um *byte* idêntico a um marcador? O receptor remove o

caractere de escape, mas mantém o *byte* seguinte, que é incorretamente interpretado como o fim do quadro. Para resolver esse problema, os caracteres de escape que fazem parte do texto também devem ser identificados com outro caractere de escape. Em outras palavras, se o caractere de escape fizer parte do texto, um caractere extra é adicionado para indicar que o segundo deles faz parte do texto. A Figura 5.5 ilustra essa situação.

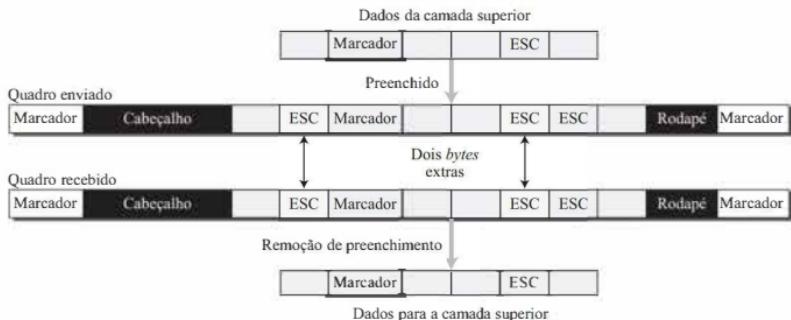



Figura 5.5 Preenchimento e remoção do preenchimento de *byte*.

> **Preenchimento de *byte* refere-se ao processo de adicionar um *byte* extra sempre que houver um marcador ou um caractere de escape no texto.**

Protocolos orientados a caracteres apresentam outro problema no que se refere à comunicação de dados. Os sistemas universais de codificação em uso atualmente, como o Unicode, usam caracteres de 16 ou 32 *bits* que conflitam com caracteres de 8 *bits*. Podemos dizer que, em geral, a tendência tem se tornado a utilização de protocolos orientados a *bits* que discutiremos a seguir.

Enquadramento orientado a *bits*

No *enquadramento orientado a bits*, a seção de dados de um quadro consiste em uma sequência de *bits* que pode ser interpretada pela camada superior como texto, gráficos, vídeo, áudio e assim por diante. No entanto, além de cabeçalhos (e possíveis rodapés), ainda precisamos de um delimitador para separar um quadro de outro. A maioria dos protocolos usa um marcador especial de 8 *bits*, 01111110, como o delimitador para determinar o início e o fim do quadro, conforme mostra a Figura 5.6.

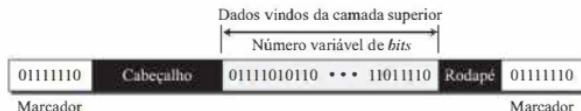



Figura 5.6 Um quadro em um protocolo orientado a *bits*.

Esse marcador pode criar o mesmo tipo de problema que vimos nos protocolos orientados a caracteres. Isto é, se o marcador aparecer nos dados, precisamos ser capazes de informar o receptor de que este não é o fim do quadro. Fazemos isso por meio da adição de um único *bit* (em vez de 1 *byte*) para evitar que essa sequência de *bits* seja confundida com um marcador. A estratégia é denominada **preenchimento de *bit***. No preenchimento de *bit*, caso seja encontrado um *bit* 0 seguido de cinco *bits* 1 consecutivos, um *bit* 0 extra é adicionado, que acaba sendo removido da seção de dados pelo receptor. Observe que o *bit* extra é adicionado após o aparecimento de um 0 seguido por cinco 1s, independentemente do valor do *bit* seguinte. Isto garante que a sequência do marcador não aparecerá inadvertidamente no quadro.

O preenchimento de *bits* refere-se ao processo de adicionar um 0 extra sempre que os dados contiverem cinco 1s consecutivos após um 0, de modo que o receptor não confunda a sequência 0111110 com um marcador.

A Figura 5.7 mostra os processos de preenchimento de *bit* no remetente e de remoção do *bit* no receptor. Perceba que, mesmo se tivermos um 0 após cinco 1s, ainda adicionamos um 0, removido pelo receptor.

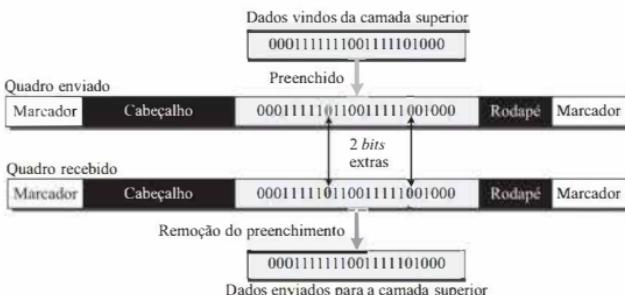



Figura 5.7 Preenchimento e remoção do preenchimento de *bit*

Isto significa que, se uma sequência semelhante a um marcador 01111110 aparecer nos dados, ela será modificada para 011111010 (preenchida) e não será confundida com um marcador pelo receptor. O marcador de fato, 01111110, não é preenchido pelo remetente, mas é reconhecido pelo receptor.

5.2.2 Controle de fluxo e erros

Definimos o controle de fluxo e de erros no Capítulo 3. Uma das responsabilidades da subcamada de controle de enlace de dados é o controle de fluxo e erros na camada de enlace de dados.

Controle de fluxo

Conforme discutimos na camada de transporte (Capítulo 3), o controle de fluxo coordena a quantidade de dados que pode ser enviada antes que uma confirmação seja recebida. Na camada de enlace de dados, o controle de fluxo é uma das tarefas da subcamada de controle de enlace de dados. O

conceito de controle de fluxo na camada de enlace de dados segue o mesmo princípio que discutimos para a camada de transporte. Embora o controle de fluxo na camada de transporte seja fim a fim (*host* a *host*), o controle de fluxo na camada de enlace de dados é nó a nó, ao longo do enlace.

Controle de erros

O controle de erros consiste tanto na detecção como na correção de erros; ele permite que o receptor informe o remetente de quaisquer quadros perdidos ou danificados durante a transmissão e também que coordene a retransmissão dos quadros pelo remetente. Na camada de enlace de dados, o termo **controle de erros** refere-se principalmente aos métodos de detecção de erros e de retransmissão. O controle de erros na camada de enlace de dados é geralmente implementado de forma simples: toda vez que for detectado um erro na transmissão, os quadros corrompidos são retransmitidos. Precisamos, entretanto, ressaltar que o controle de erros na camada de transporte é fim a fim, mas o controle de erros na camada de enlace de dados é nó a nó, ao longo do enlace. Em outras palavras, cada vez que um quadro passa por um enlace, precisamos ter certeza de que ele não está corrompido.

5.2.3 Detecção e correção de erros

Na camada de enlace de dados, se um quadro for corrompido entre os dois nós, ele precisa ser corrigido antes de continuar sua jornada até os outros nós. No entanto, a maioria dos protocolos da camada de enlace simplesmente descarta o quadro e deixa que os protocolos da camada superior lidem com a retransmissão dele. Alguns protocolos sem fio, entretanto, tentam corrigir o quadro corrompido.

Introdução

Primeiramente, discutiremos algumas questões relacionadas, direta ou indiretamente, à detecção e correção de erros.

Tipos de erros

Sempre que os *bits* trafegam de um ponto a outro, eles estão sujeitos a modificações imprevisíveis causadas por **interferências**, que podem alterar a forma do sinal. O termo **erro de um único *bit*** significa que apenas um *bit* de uma determinada unidade de dados (por exemplo, um *byte*, um caractere ou um pacote) é alterado de 1 para 0 ou de 0 para 1. O termo **erro em rajada** significa que 2 ou mais *bits* na unidade de dados são alterados de 1 para 0 ou de 0 para 1. A Figura 5.8 mostra os efeitos de um erro de um único *bit* e de um erro em rajada sobre uma unidade de dados.

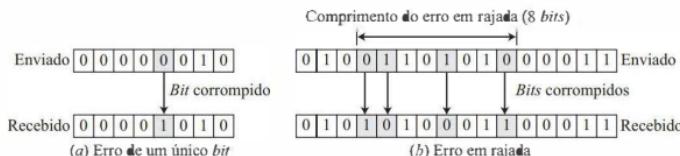



Figura 5.8 Erro de um único *bit* e erro em rajada.

Um erro em rajada é mais provável de ocorrer do que um erro de um único *bit*, porque a duração do sinal de ruído é normalmente maior que a duração de 1 *bit*, isto é, quando o ruído afeta os dados, ele afeta um conjunto de *bits*. O número de *bits* afetados depende da taxa de transmissão de dados e da duração do ruído. Por exemplo, se estivermos enviando dados a 1 kbps, um ruído com

duração de 1/100 segundos pode afetar 10 *bits*; se estivermos enviando dados a 1 Mbps, o mesmo ruído pode afetar 10.000 *bits*.

Redundância

O conceito central da detecção ou correção de erros é a *redundância*. Para sermos capazes de detectar ou corrigir erros, precisamos enviar alguns *bits* extras juntamente com os dados. Estes *bits* redundantes são adicionados pelo remetente e removidos pelo receptor. A sua presença permite que o receptor detecte ou corrija *bits* corrompidos.

Detecção *versus* correção

A correção de erros é mais difícil que a sua detecção. Na *detecção de erros*, estamos verificando apenas se algum erro ocorreu; nesse caso, a resposta é simplesmente sim ou não. Não estamos nem mesmo interessados no número de *bits* corrompidos. Não interessa saber se ocorreu um erro de um único *bit* ou um erro em rajada. Já na *correção de erros*, precisamos saber o número exato de *bits* que foram corrompidos e, ainda mais importante, a sua localização na mensagem. O número de erros e o tamanho da mensagem são fatores importantes. Se precisarmos corrigir um único erro em uma unidade de dados de oito *bits*, precisamos considerar oito possíveis locais de erro; se precisarmos corrigir dois erros em uma unidade de dados do mesmo tamanho, precisamos considerar 28 possibilidades (combinação de oito posições tomadas duas a duas). Não é difícil imaginar a dificuldade do receptor em encontrar 10 erros em uma unidade de dados de 1.000 *bits*. A seguir, nós nos concentraremos na detecção de erros; a correção de erros é mais difícil, mas será discutida brevemente no Capítulo 8.

Codificação

A redundância é obtida por meio de diversos esquemas de codificação. O remetente adiciona *bits* redundantes por meio de um processo que cria uma relação entre os *bits* redundantes e os *bits* de dados reais. O receptor verifica a relação entre os dois conjuntos de *bits* para detectar erros. A razão entre o número de *bits* redundantes e de *bits* de dados, bem como a robustez do processo, são fatores importantes em qualquer esquema de codificação.

Podemos dividir os esquemas de codificação em duas grandes categorias: *codificação de bloco* e *codificação convolucional*. Neste livro, nós nos concentramos na codificação de bloco; a codificação convolucional é mais complexa e está além do escopo desta obra.

Codificação de bloco

Na codificação de bloco, dividimos a mensagem em blocos, cada um tendo k *bits*, chamados **palavras de dados**. Adicionamos r *bits* redundantes a cada bloco para obter o comprimento $n = k + r$. Os blocos de n *bits* resultantes são denominados **palavras de código**. A forma como os r *bits* extras são escolhidos ou calculados é algo que discutiremos mais adiante. Por ora, é importante saber que temos um conjunto de palavras de dados, cada uma de tamanho k, e um conjunto de palavras de código, cada uma de tamanho n. Com k *bits*, podemos criar uma combinação de 2^k palavras de dados; com n *bits*, podemos criar uma combinação de 2^n palavras de código. Como $n > k$, o número de palavras de código possíveis é maior que o número de palavras de dados possíveis. O processo de codificação de bloco é um para um; uma determinada palavra de dados é sempre codificada como a mesma palavra de código. Isso significa que temos $2^n - 2^k$ palavras de código que não são utilizadas. Dizemos que essas palavras de código são inválidas ou ilegais. O truque na detecção de erros depende da existência desses códigos inválidos, conforme discutiremos a seguir. Se o receptor receber uma palavra de código inválida, isto indica que os dados foram corrompidos durante a transmissão.

Detecção de erros

Como os erros podem ser detectados usando codificação de bloco? Se as duas condições seguintes forem satisfeitas, o receptor pode detectar uma alteração na palavra de código original.

1. O receptor tem (ou pode determinar) uma lista de palavras de código válidas.
2. A palavra de código original foi transformada em uma palavra de código inválida.

A Figura 5.9 mostra o papel da codificação de bloco na detecção de erros.

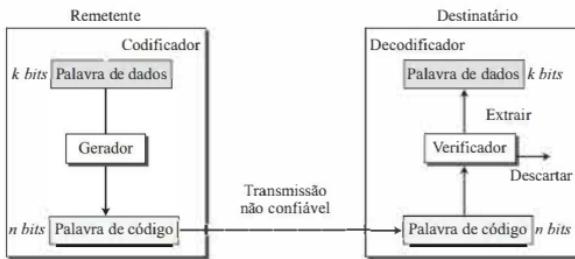



Figura 5.9 Processo de detecção de erros na codificação de bloco.

O remetente cria palavras de código a partir de palavras de dados usando um gerador que aplica as regras e procedimentos de codificação (discutidos mais adiante). Cada palavra de código enviada para o receptor pode vir a ser modificada durante a transmissão. Se a palavra de código recebida corresponde a uma palavra de código válida, a palavra é aceita; neste caso, a palavra de código correspondente é extraída e usada. Se a palavra de código recebida não for válida, ela é descartada. No entanto, se a palavra de código for corrompida durante a transmissão e a palavra recebida ainda corresponder a uma palavra de código válida, o erro passa despercebido.

Exemplo 5.1

Consideremos que $k = 2$ e $n = 3$. A Tabela 5.1 mostra a lista de palavras de dados e palavras de código. Mais adiante, veremos como obter uma palavra de código a partir de uma palavra de dados.

Tabela 5.1 Um código para a detecção de erros do Exemplo 5.1.

Palavras de dados	Palavras de código	Palavras de dados	Palavras de código
00	000	10	101
01	011	11	110

Considere que o remetente codifica a palavra de dados 01 como 011 e a envia para o receptor. Considere os seguintes casos:

1. O receptor recebe 011, uma palavra de código válida. O receptor extrai a palavra de dados 01 a partir dela.
2. A palavra de código é corrompida durante a transmissão, e o valor 111 é recebido (o *bit* mais à esquerda foi corrompido). Não é uma palavra de código válida, portanto, ela é descartada.
3. A palavra de código é corrompida durante a transmissão, de modo que o valor recebido é 000 (os dois *bits* da direita foram corrompidos). É uma palavra de código válida. O receptor extrai incorretamente a palavra de dados 00. Os dois *bits* corrompidos tornaram o erro indetectável.

Um código de detecção de erros pode detectar apenas os tipos de erros para os quais ele foi projetado; outros tipos de erros podem passar despercebidos.

Distância de Hamming

Um dos conceitos centrais da codificação voltada ao controle de erros é a ideia da **distância de Hamming**. A distância de Hamming entre duas palavras (do mesmo tamanho) corresponde ao número de diferenças entre os *bits* correspondentes. Representamos a distância de Hamming entre duas palavras x e y como $d(x, y)$. Podemos nos perguntar por que a distância de Hamming é importante para a detecção de erros. A razão é que a distância de Hamming entre a palavra de código recebida e a palavra de código enviada equivale ao número de *bits* que foram corrompidos durante a transmissão. Por exemplo, se a palavra de código 00000 é enviada e 01101 é recebida, 3 *bits* foram corrompidos e a distância de Hamming entre as duas palavras é $d(00000, 01101) = 3$. Ou seja, se a distância de Hamming entre as palavras de código enviadas e recebidas não for zero, a palavra de código foi corrompida durante a transmissão.

A distância de Hamming pode ser facilmente encontrada se aplicarmos a operação de XOR (também conhecida como OU-Exclusivo e representada como ⊕) sobre as duas palavras e contar o número de 1s no resultado. Perceba que a distância de Hamming é um valor maior ou igual a zero.

A distância de Hamming entre duas palavras é o número de diferenças entre *bits* correspondentes.

Exemplo 5.2

Determinaremos a distância de Hamming entre dois pares de palavras.

1. A distância de Hamming $d(000, 011)$ é 2 porque $(000 \oplus 011)$ é 011 (dois 1s).
2. A distância de Hamming $d(10101, 11110)$ é 3 porque $(10101 \oplus 11110)$ é 01011 (três 1s).

Distância de Hamming mínima para a detecção de erros

Em um conjunto de palavras de código, a distância de Hamming mínima corresponde à menor distância de Hamming entre todos os pares possíveis de palavras de código. Agora, determinaremos a distância de Hamming mínima de um código para que ele seja capaz de detectar até s erros. Se s erros ocorrerem durante a transmissão, a distância de Hamming entre as palavras de código enviadas e recebidas é s. Se o nosso sistema deve detectar até s erros, a distância mínima entre as palavras de código válidas deve ser $(s + 1)$, de modo que a palavra de código recebida não corresponda a uma palavra de código válida. Em outras palavras, se a distância mínima entre todas as palavras de código válidas for $(s + 1)$, a palavra de código recebida não pode ser erroneamente confundida com outra palavra de código. O erro será detectado. Precisamos esclarecer um ponto aqui: apesar de um código com $d_{min} = s + 1$ ser capaz de detectar mais do que s erros em alguns casos especiais, quando s ou menos erros ocorrem, tem-se a garantia de que eles serão detectados.

Para garantir a detecção de até *s* erros em todos os casos, a distância de Hamming mínima em um código de bloco deve ser $d_{min} = s + 1$.

Podemos observar esse critério geometricamente. Suponha que a palavra de código enviada x esteja no centro de um círculo de raio s. Todas as palavras de código recebidas criadas por 0 a s erros correspondem a pontos no interior do círculo ou no perímetro do círculo. Todas as outras palavras de código válidas devem estar fora do círculo, conforme mostra a Figura 5.10. Isto significa que d_{min} deve ser um número inteiro maior que s $d_{min} = s + 1$.

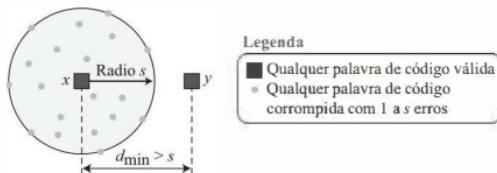



Figura 5.10 Conceito geométrico explicando d_{min} no contexto de detecção de erros.

Exemplo 5.3

A distância de Hamming mínima para o nosso primeiro esquema de codificação (Tabela 5.1) é 2. Este código garante apenas a detecção de um único erro. Por exemplo, se a terceira palavra de código (101) for enviada e um erro ocorrer, a palavra de código recebida não corresponde a uma palavra de código válida. Caso ocorram dois erros, no entanto, a palavra de código recebida pode corresponder a uma palavra de código válida e os erros não serão detectados.

Exemplo 5.4

Um esquema de codificação tem uma distância de Hamming $d_{min} = 4$. Esse código garante a detecção de até três erros ($d = s + 1$ ou $s = 3$).

Códigos de bloco lineares

Quase todos os códigos de bloco usados atualmente pertencem a um subconjunto de códigos de bloco denominados *códigos de bloco lineares*. O uso de códigos de bloco não lineares para detecção e correção de erros não é tão generalizado porque a estrutura deles dificulta a sua análise teórica e implementação. Portanto, vamos nos concentrar nos códigos de bloco lineares, cuja definição requer o conhecimento de álgebra abstrata (particularmente corpos de Galois), algo que está além do escopo deste livro. Portanto, apresentamos uma definição informal. Para nossos propósitos, um código de bloco linear é um código em que a operação de XOR (adição módulo 2) de duas palavras de código válidas resulta em uma outra palavra de código válida.

Exemplo 5.5

O código da Tabela 5.1 é um código de bloco linear porque o resultado de aplicar a operação de XOR sobre qualquer par de palavras de código resulta em uma palavra de código válida. Por exemplo, fazer o XOR da segunda com a terceira palavras de código cria a quarta palavra de código.

Distância mínima em códigos de bloco lineares

É simples calcular a distância de Hamming mínima de um código de bloco linear. A distância de Hamming mínima corresponde ao número de 1s na palavra de código válida com o menor número de 1s, porém diferente de zero.

Exemplo 5.6

No nosso primeiro código (Tabela 5.1), os números de 1s nas palavras de código diferentes de zero são 2, 2 e 2. Assim, a distância de Hamming mínima é $d_{min} = 2$.

Código de paridade

Talvez o código de detecção de erros mais conhecido seja o **código de paridade**, que é um código de bloco linear. Nele, uma palavra de dados de *k bits* é transformada em uma palavra de código de *n bits* onde $n = k + 1$. O *bit* extra, denominado *bit* de paridade, é selecionado de modo a fazer com que o número total de 1s na palavra de código seja par. Embora algumas implementações especifiquem um número ímpar de 1s, discutimos apenas o caso par. A distância de Hamming mínima para essa categoria é $d_{min} = 2$, o que significa que o código é detector de erros de um único *bit*. Nosso primeiro código (Tabela 5.1) é de paridade ($k = 2$ e $n = 3$). O código da Tabela 5.2 também é de paridade com $k = 4$ e $n = 5$.

Tabela 5.2 Código de paridade simples C(5, 4).

Palavras de dados	Palavras de código	Palavras de dados	Palavras de código
0000	00000	1000	10001
0001	00011	1001	10010
0010	00101	1010	10100
0011	00110	1011	10111
0100	01001	1100	11000
0101	01010	1101	11011
0110	01100	1110	11101
0111	01111	1111	11110

A Figura 5.11 mostra uma possível estrutura de um codificador (no remetente) e de um decodificador (no destinatário).

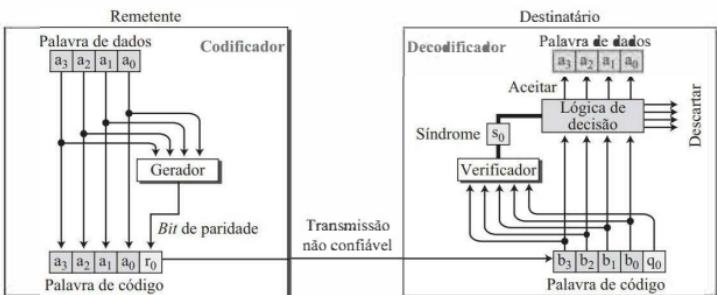



Figura 5.11 Codificador e decodificador para um código de paridade simples.

O codificador usa um gerador que toma uma cópia de uma palavra de dados de 4 *bits* (a_0, a_1, a_2 e a_3) e gera um *bit* de paridade r_0. Os *bits* da palavra de dados e o *bit* de paridade criam a palavra de código de 5 *bits*. O *bit* de paridade adicionado faz com que o número de 1s na palavra de código

seja par. Isso é normalmente obtido por meio da soma dos 4 *bits* da palavra de dados (módulo 2); o resultado é o *bit* de paridade. Em outras palavras,

$$r_0 = a_3 + a_2 + a_1 + a_0 \qquad \textbf{(módulo 2)}$$

Se o número de 1s for par, o resultado é 0; se for ímpar, o resultado é 1. Em ambos os casos, o número total de 1s na palavra de código é par.

O remetente envia a palavra de código, a qual pode ser corrompida durante a transmissão. O destinatário recebe uma palavra de 5 *bits*. O verificador no receptor faz a mesma coisa que o gerador no remetente, mas com uma exceção: a adição é feita considerando todos os 5 *bits*. O resultado, chamado **síndrome**, corresponde a apenas 1 *bit*. A síndrome é 0 quando o número de 1s na palavra de código recebida for par; caso contrário, a síndrome é 1.

$$s_0 = b_3 + b_2 + b_1 + b_0 + q_0 \qquad \textbf{(módulo 2)}$$

A síndrome é passada para o analisador de lógica de decisão. Se a síndrome for 0, não há erro detectável na palavra de código recebida; a parte de dados da palavra de código recebida é aceita como palavra de dados. Já se a síndrome for 1, a parte de dados recebida da palavra de código é descartada. A palavra de dados não é criada.

Exemplo 5.7

Vejamos alguns cenários de transmissão. Considere que o remetente envia a palavra de dados 1011. A palavra de código criada a partir dessa palavra de dados é 10111, enviada para o receptor. Examinemos cinco casos:

1. Nenhum erro ocorre; a palavra de código recebida é 10111. A síndrome é 0. A palavra de dados 1011 é criada.
2. Um erro de um único *bit* altera a_1. A palavra de código recebida é 10011. A síndrome é 1. Nenhuma palavra de dados é criada.
3. Um erro de um único *bit* altera r_0. A palavra de código recebida é 1011**0**. A síndrome é 1. Nenhuma palavra de dados é criada. Observe que, embora nenhum dos *bits* da palavra de dados esteja corrompido, nenhuma palavra de dados é criada porque o código não é suficientemente sofisticado para mostrar a posição do *bit* corrompido.
4. Um erro altera r_0 e um segundo erro altera a_3. A palavra de código recebida é **0**011**0**. A síndrome é 0. A palavra de dados 0011 é criada no receptor. Perceba que, aqui, a palavra de dados é erroneamente criada devido ao valor da síndrome. O decodificador de paridade simples não é capaz de detectar um número par de erros. Os erros se anulam mutuamente e levam a um valor de síndrome igual a 0.
5. Três *bits* – a_3, a_2 e a_1 – são alterados devido a erros. A palavra de código recebida é **010**11. A síndrome é 1. A palavra de dados não é criada. Isto mostra que a verificação de paridade simples, que garante a detecção de um único erro, pode também detectar qualquer número impar de erros.

Um código de paridade é capaz de detectar um número ímpar de erros.

Códigos cíclicos

Códigos cíclicos são um tipo especial de códigos de bloco lineares que possuem uma propriedade extra. Em um **código cíclico**, se uma palavra de código for deslocada ciclicamente (rotacionada), o resultado é outra palavra de código. Por exemplo, se 1011000 for um palavra de código, podemos deslocá-la ciclicamente para a esquerda e obter 0110001, que também é uma palavra de código. Nesse caso, se nomearmos os *bits* da primeira palavra de a_0 a a_6 e os *bits* na segunda palavra de b_0 a b_6, podemos deslocar os *bits* conforme segue:

$$b_1 = a_0 \quad b_2 = a_1 \quad b_3 = a_2 \quad b_4 = a_3 \quad b_5 = a_4 \quad b_6 = a_5 \quad b_0 = a_6$$

Na equação mais à direita, o último *bit* da primeira palavra é rotacionado ciclicamente e torna-se o primeiro *bit* da segunda palavra.

Verificação de Redundância Cíclica

Podemos criar códigos cíclicos para corrigir erros. No entanto, o embasamento teórico necessário está além do escopo deste livro. Nesta seção, discutimos simplesmente um subconjunto de códigos cíclicos denominados **Verificação de Redundância Cíclica** (CRC – Cyclic Redundancy Check), que são usados em redes como LANs e WANs.

A Tabela 5.3 mostra um exemplo de um código CRC. Podemos observar as propriedades lineares e cíclicas desse código.

Tabela 5.3 Um código CRC com C(7, 4).

Palavra de dados	Palavra de código	Palavra de dados	Palavra de código
0000	0000000	1000	1000101
0001	0001011	1001	1001110
0010	0010110	1010	1010011
0011	0011101	1011	1011000
0100	0100111	1100	1100010
0101	0101100	1101	1101001
0110	0110001	1110	1110100
0111	0111010	1111	1111111

A Figura 5.12 mostra uma possível estrutura para o codificador e para o decodificador.

No codificador, a palavra de dados tem k *bits* (aqui, $k = 4$); a palavra de código tem n *bits* (aqui, $n = 7$). O tamanho da palavra de dados é expandido pela adição de $n - k$ (aqui, 3) 0s à direita da palavra. O resultado de n *bits* é alimentado no gerador, que usa um divisor de tamanho $n - k + 1$ (aqui, 4), predefinido e pré-acordado. O gerador divide a palavra de dados expandida pelo divisor (divisão módulo 2). O quociente da divisão é descartado, o resto ($r_2r_1r_0$) é concatenado com a palavra de dados para criar a palavra de código.

O decodificador recebe a palavra de código (possivelmente corrompida durante a transmissão). Uma cópia de todos os n *bits* é alimentada no verificador, que é uma réplica do gerador. O resto produzido pelo verificador é uma síndrome de $n - k$ (aqui, 3) *bits*, fornecida ao analisador da lógica de decisão. Se os *bits* da síndrome forem todos 0s, os 4 *bits* mais à esquerda da palavra de código são considerados a palavra de dados (interpretada como livre de erros); caso contrário, os 4 *bits* são descartados (erro).

Codificador Analisemos o codificador mais de perto. Ele recebe uma palavra de dados e a expande com um número de 0s igual a $n - k$. Em seguida, ele divide a palavra de dados expandida pelo divisor, conforme mostra a Figura 5.13.

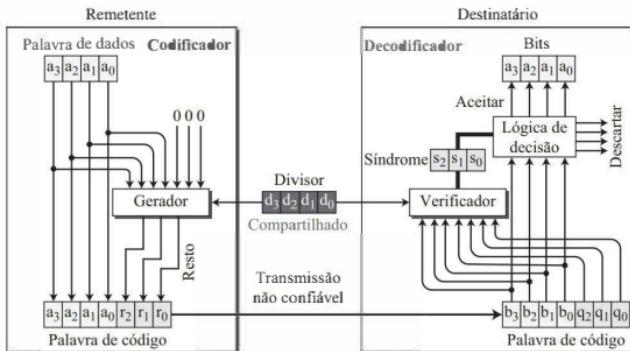



Figura 5.12 Codificador e decodificador CRC.

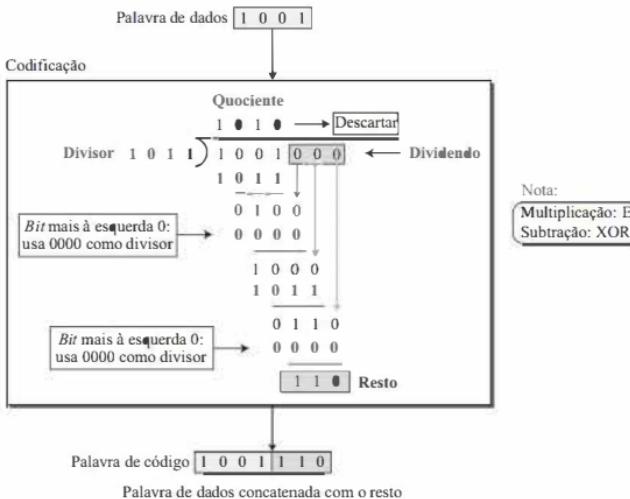



Figura 5.13 Divisão no codificador CRC.

O processo de divisão binária módulo 2 é equivalente ao processo de divisão geralmente utilizado para números decimais. Entretanto, nesse caso, o mesmo não vale para a adição e a subtração; usamos a operação XOR para efetuar ambas as operações.

Assim como na divisão decimal, o processo é feito passo a passo. Em cada passo, é feito o XOR de uma cópia do divisor com os 4 *bits* do dividendo. O resultado da operação de XOR (resto) tem 3 *bits* (nesse caso), que são usados no passo seguinte após um *bit* extra ser trazido para baixo, de

modo a deixá-lo com um comprimento de 4 *bits*. Há um ponto importante que precisamos lembrar nesse tipo de divisão: se o *bit* mais à esquerda do dividendo (ou a parte utilizada em cada passo) for 0, o passo não pode utilizar o divisor normal; precisamos usar um divisor composto apenas por 0s.

Quando não há *bits* para trazer para baixo, temos o resultado final. O resto, de 3 *bits*, corresponde aos *bits* de verificação (r_2, r_1 e r_0). Ele é concatenado à palavra de dados para criar a palavra de código.

Decodificador A palavra de código pode ser alterada durante a transmissão. O decodificador executa o mesmo processo de divisão realizado no codificador. O resto da divisão corresponde à síndrome. Se ela for composta apenas por 0s, existe uma elevada probabilidade de não ter havido erros; a palavra de dados é separada da palavra de código recebida e aceita. Caso contrário, todos os *bits* são descartados. A Figura 5.14 mostra dois casos: a figura do lado esquerdo mostra o valor da síndrome quando não há erros; a síndrome, nesse caso, é 000. O lado direito da figura mostra o caso em que há erro em um único *bit*. A síndrome não é composta apenas por 0s (ela vale 011).

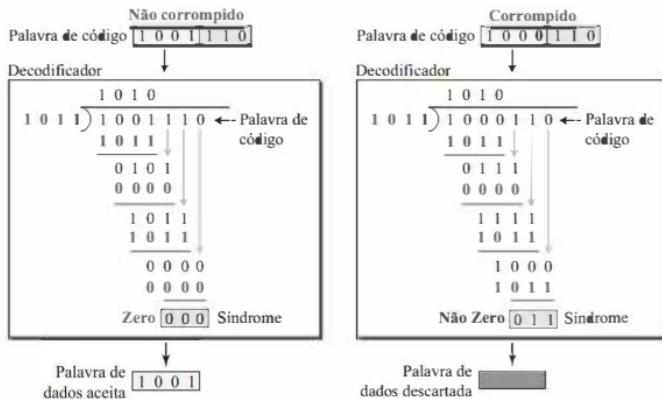



Figura 5.14 Divisão no decodificador CRC em duas situações.

Divisor Podemos estar nos perguntando como o divisor 1011 foi escolhido. Isto depende do que se espera do código. Discutimos os critérios no *site* www.grupoa.com.br. Alguns dos divisores-padrão utilizados na área de redes são mostrados na Tabela 5.4. O número no nome do divisor (por exemplo, CRC-32) refere-se ao grau do polinômio (a sua maior potência) que representa o divisor. O número de *bits* é sempre uma unidade a mais do que o grau do polinômio. Por exemplo, o CRC-8 tem 9 *bits* e o CRC-32 tem 33 *bits*.

Tabela 5.4 Polinômios-padrão.

Nome	Binário	Aplicação
CRC-8	100000111	Cabeçalho ATM
CRC-10	11000110101	ATM AAL
CRC-16	10001000000100001	HDLC
CRC-32	100000100110000010001110110110111	LANs

Polinômios

Uma maneira melhor de entender os códigos cíclicos e a forma como eles podem ser analisados é representá-los como polinômios. Discutimos polinômios no *site* www.grupoa.com.br para os leitores interessados.

Requisitos

Podemos comprovar matematicamente que uma sequência de *bits* precisa ter pelo menos duas propriedades para ser considerada um gerador (divisor):

1. A sequência deve ter pelo menos dois *bits*.
2. Os *bits* mais à direita e mais à esquerda devem ser ambos 1s.

Eficácia

Podemos comprovar matematicamente a eficácia do CRC conforme mostrado a seguir.

- **Erros individuais.** Todos os geradores que satisfazem os requisitos anteriores são capazes de detectar qualquer erro de um único *bit*.
- **Número ímpar de erros.** Todos os geradores que satisfazem os requisitos anteriores são capazes de detectar qualquer número ímpar de erros, contanto que o gerador seja divisível por $(11)_2$ usando divisão binária em aritmética módulo 2; caso contrário, apenas alguns erros em um número ímpar de *bits* serão detectados.
- **Erros em rajada.** Se considerarmos que o comprimento do erro em rajada é de L *bits* e que r é o comprimento do resto (r é o comprimento do gerador menos 1; ele é também o valor da potência mais elevada no polinômio que representa o gerador):
  a. Todos os erros em rajada de tamanho $L \leq r$ são detectados.
  b. Todos os erros em rajada de tamanho $L = r + 1$ são detectados com probabilidade $1 - (0{,}5)^{r-1}$.
  c. Todos os erros em rajada de tamanho $L > r + 1$ são detectados com probabilidade $1 - (0{,}5)^{r}$.

Vantagens de códigos cíclicos

Os códigos cíclicos podem ser facilmente implementados em *hardware* e *software*. Eles são especialmente rápidos quando implementados em *hardware*, o que tornou os códigos cíclicos bons candidatos para serem usados em muitas redes. No *site* www.grupoa.com.br, mostramos como a divisão pode ser feita usando um registrador de deslocamento que está incluído no *hardware* do nó.

Soma de verificação

A soma de verificação, ou *checksum*, é uma técnica de detecção de erros que pode ser aplicada a uma mensagem de qualquer tamanho. Na Internet, a técnica de soma de verificação é usada principalmente nas camadas de rede e de transporte e não na camada de enlace de dados. Entretanto, para tornar a nossa discussão sobre técnicas de detecção de erros mais completa, discutimos a soma de verificação (*checksum*) neste capítulo.

Na origem, a mensagem é primeiramente dividida em unidades de *m bits*. O gerador cria, então, uma unidade extra de *m bits* denominada *soma de verificação*, enviada com a mensagem. No destino, o verificador cria uma nova soma de verificação a partir da combinação da mensagem e da soma de verificação recebida. Se a nova soma de verificação for composta apenas por 0s, a mensagem é aceita; caso contrário, a mensagem é descartada (Figura 5.15). Perceba que, em uma aplicação real, a unidade da soma de verificação não é necessariamente adicionada ao final da mensagem; ela pode ser inserida no meio da mensagem.

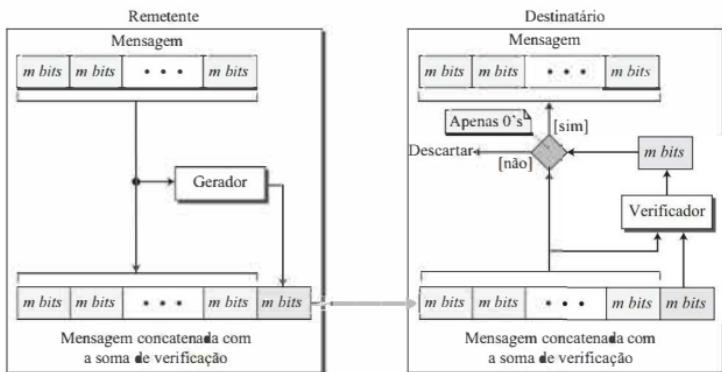



Figura 5.15 Soma de verificação (*checksum*).

Conceitos

A ideia por trás da soma de verificação tradicional é simples. Mostramos isto usando um exemplo simples.

Exemplo 5.8

Considere que a mensagem seja uma lista de cinco números de 4 *bits* que queremos enviar a um destino. Além de enviar esses números, enviamos também a soma deles. Por exemplo, se o conjunto de números for (7, 11, 12, 0, 6), enviamos (7, 11, 12, 0, 6, **36**), onde 36 é a soma dos números originais. O receptor soma os cinco números e compara o resultado com a soma recebida. Se os dois resultados forem iguais, o receptor considera que não houve qualquer erro, aceita os cinco números e então descarta a soma. Caso contrário, há um erro em algum lugar e a mensagem não é aceita.

Adição em complemento de um O exemplo anterior apresenta um grande problema. Cada número pode ser escrito como uma palavra de 4 *bits* (todos eles são menores do que 15), exceto pela soma. Uma solução é usar aritmética em **complemento de um**. Nesse tipo de aritmética, podemos representar números sem sinal entre 0 e $2^m - 1$ usando apenas *m bits*. Se o número tem mais do que *m bits*, os *bits* extras mais à esquerda precisam ser somados aos *m bits* mais à direita.

Exemplo 5.9

No exemplo anterior, o número decimal 36 em binário é escrito como $(100100)_2$. Para transformá-lo em um número de 4 *bits*, somamos o *bit* extra mais à esquerda aos quatro *bits* da direita, conforme mostrado a seguir.

$$(10)_2 + (0100)_2 = (0110)_2 \rightarrow (6)_{10}$$

Em vez de enviar 36 como o valor da soma, podemos enviar o valor 6, ou seja, (7, 11, 12, 0, 6, **6**). O destinatário pode somar os cinco primeiros números usando aritmética em complemento de um. Se o resultado for 6, os números são aceitos; caso contrário, eles são rejeitados.

Soma de verificação Podemos facilitar o trabalho do destinatário se enviarmos o complemento da soma, conhecido como soma de verificação, ou *checksum*. Na aritmética em complemento de um, o complemento de um número é encontrado invertendo-se todos os *bits* (transformando todos os 1s em 0s e todos os 0s em 1s). Isto equivale a subtrair o número de $2^m - 1$. Na aritmética em complemento de um, temos dois 0s: um positivo e um negativo, que são o complemento um do outro. O zero positivo tem todos os *m bits* valendo 0; o zero negativo tem todos os *bits* valendo 1 (ele corresponde a $2^m - 1$). Se somarmos um número ao seu complemento, obtemos um zero negativo (um número com todos os *bits* valendo 1). Quando o destinatário soma todos os cinco números (incluindo a soma de verificação), ele obtém um zero negativo. O destinatário pode complementar o resultado novamente para obter um zero positivo.

Exemplo 5.10

Apliquemos a ideia da soma de verificação no Exemplo 5.9. O remetente soma todos os cinco números usando aritmética em complemento de um para obter a soma = 6. O remetente, então, complementa o resultado para obter a soma de verificação = 9, que é o resultado de 15 – 6. Observe que $6 = (0110)_2$ e $9 = (1001)_2$, que são complementos um do outro. O remetente envia os cinco números (dados) juntamente com a soma de verificação (7, 11, 12, 0, 6, **9**). Se não houver qualquer corrupção durante a transmissão, o destinatário recebe (7, 11, 12, 0, 6, **9**) e soma esses números usando aritmética em complemento de um para obter o valor 15. O remetente complementa o valor 15 para obter 0. Isso mostra que os dados não foram corrompidos. A Figura 5.16 ilustra o processo.

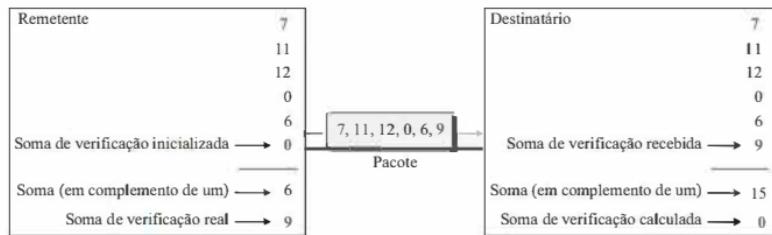
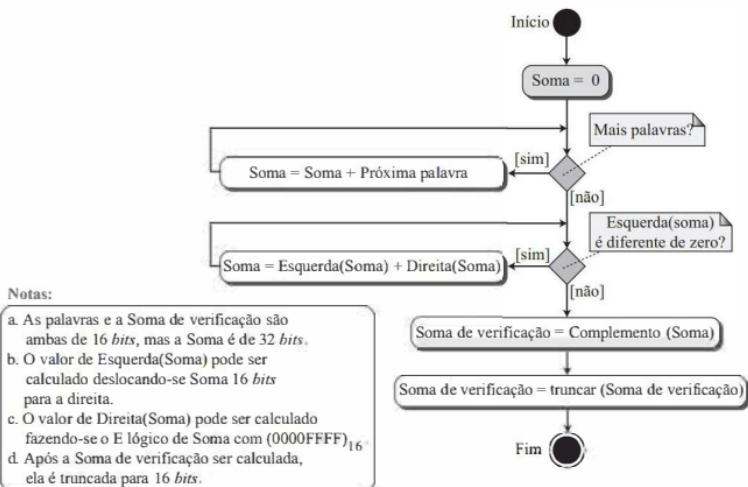



Figura 5.16 Esquema para o Exemplo 5.10.

Soma de verificação na Internet

Tradicionalmente, a Internet usa uma soma de verificação de 16 *bits*. O remetente e o destinatário seguem os passos descritos na Tabela 5.5. São cinco os passos usados tanto pelo remetente como pelo destinatário.

Algoritmo

Podemos usar o diagrama de fluxo da Figura 5.17 para mostrar o algoritmo do cálculo da soma de verificação. Um programa em qualquer linguagem pode ser facilmente escrito com base nesse algoritmo. Observe que o primeiro laço apenas calcula a soma das unidades de dados em complemento de dois; o segundo laço rotaciona os *bits* extras criados pelo cálculo em complemento de dois para simular os cálculos em complemento de um. Isto é necessário porque quase todos os computadores atuais fazem cálculos em complemento de dois.

Tabela 5.5 Procedimento para calcular a soma de verificação tradicional.

Remetente	Destinatário
1. A mensagem é dividida em palavras de 16 *bits*.	1. A mensagem e a soma de verificação são recebidas.
2. O valor da soma de verificação é inicialmente fixado em zero.	2. A mensagem é dividida em palavras de 16 *bits*.
3. Todas as palavras, incluindo a soma de verificação, são somadas usando soma em complemento de um.	3. Todas as palavras são somadas usando soma em complemento de um.
4. A soma é complementada e torna-se a soma de verificação.	4. A soma é complementada e torna-se a soma de verificação.
5. A soma de verificação é enviada juntamente com os dados.	5. Se o valor da soma de verificação for 0, a mensagem é aceita; caso contrário, ela é rejeitada.

Figura 5.17 Algoritmo para calcular uma soma de verificação tradicional.

Eficácia A soma de verificação tradicional usa um número pequeno de *bits* (16) para detectar erros em uma mensagem de qualquer tamanho (algumas vezes milhares de *bits*). No entanto, essa técnica não é tão efetiva quanto o CRC com relação à sua capacidade de verificar erros. Por exemplo, se o valor de uma palavra é incrementado e o valor de outra palavra é decrementado da mesma quantidade, os dois erros não podem ser detectados porque a soma de verificação permanece a mesma. Além disso, se os valores de várias palavras forem incrementados, mas a sua soma e a soma de verificação não se alterarem, os erros não serão detectados. Fletcher e Adler propuseram algumas somas de verificação ponderadas que eliminam o primeiro desses problemas. No entanto, a tendência na Internet, particularmente no projeto de novos protocolos, é substituir a soma de verificação pelo CRC.

Outras abordagens para a soma de verificação

Conforme mencionado anteriormente, existe um grande problema com o cálculo da soma de verificação tradicional. Se dois itens de dados de 16 *bits* forem invertidos durante a transmissão, a soma de verificação não será capaz de detectar esse erro. A razão é que a soma de verificação tradicional não é ponderada: ela trata cada item de dados de forma idêntica. Em outras palavras, a ordem dos itens de dados é irrelevante para o cálculo. Várias abordagens podem ser utilizadas para evitar esse problema. Mencionamos duas delas aqui: Fletcher e Adler.

Soma de verificação de Fletcher A soma de verificação de Fletcher foi concebida para atribuir um peso a cada item de dados de acordo com sua posição. Fletcher propôs dois algoritmos: um de 8 e outro de 16 *bits*. O primeiro, Fletcher de 8 *bits*, faz os cálculos sobre itens de dados de 8 *bits* e gera uma soma de verificação de 16 *bits*. O segundo, Fletcher de 16 *bits*, faz os cálculos sobre itens de dados de 16 *bits* e gera uma soma de verificação de 32 *bits*.

O Fletcher de 8 *bits* é calculado sobre octetos (*bytes*) de dados e gera uma soma de verificação de 16 *bits*. O cálculo é feito em módulo 256 (2^8), o que significa que os resultados intermediários são divididos por 256 e o resto é mantido. O algoritmo utiliza dois acumuladores, L e R. O primeiro simplesmente soma itens de dados uns aos outros; o segundo introduz o peso no cálculo. Existem diversas variantes do algoritmo de Fletcher de 8 *bits*; mostramos uma variante simples na Figura 5.18.

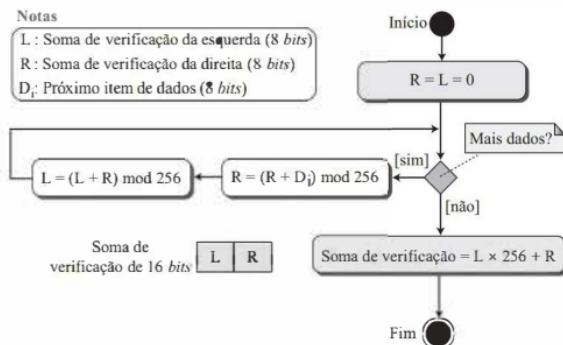



Figura 5.18 Algoritmo para calcular uma soma de verificação de Fletcher de 8 *bits*.

A soma de verificação Fletcher de 16 *bits* é semelhante à soma de verificação Flecher de 8 *bits*, mas é calculada sobre itens de dados de 16 *bits* e gera uma soma de verificação de 32 *bits*. O cálculo é feito em módulo 65.536.

Soma de verificação de Adler A soma de verificação de Adler é uma soma de verificação de 32 *bits*. A Figura 5.19 mostra um algoritmo simples na forma de um fluxograma. Essa abordagem é semelhante à abordagem de Fletcher de 16 *bits*, com três diferenças. Primeiro, o cálculo é realizado em *bytes* individuais em vez de 2 *bytes* de cada vez. Segundo, o módulo é um número primo (65.521) em vez de 65.536. Terceiro, L é inicializado com o valor 1 em vez de 0. Prova-se que um módulo primo tem uma melhor capacidade de detecção em algumas combinações de dados.

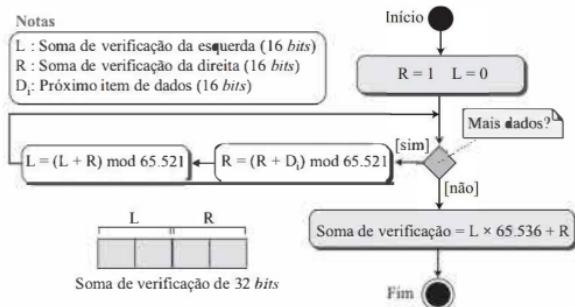



Figura 5.19 Algoritmo para calcular a soma de verificação de Adler.

5.2.4 Dois protocolos de DLC

Agora que terminamos a discussão de todas as questões relacionadas à subcamada DLC, discutimos dois protocolos DLC que de fato implementam tais conceitos. O primeiro, denominado HDLC, é a base de muitos protocolos que foram projetados para LANs. O segundo, conhecido como PPP, é um protocolo derivado do HDLC e é usado para enlaces ponto a ponto.

HDLC

O **Controle de Enlace de Dados de Alto Nível** (HDLC – High-level Data Link Control) é um protocolo orientado a *bits* para comunicação através de enlaces ponto a ponto ou multiponto. Ele implementa o protocolo *Stop-and-Wait* que discutimos no Capítulo 3.

Configurações e modos de transferência

O HDLC fornece dois modos comuns de transferência que podem ser usados em diferentes configurações: *Modo de Resposta Normal* (NRM – Normal Response Mode) e *Modo Balanceado Assíncrono* (ABM – Asynchronous Balanced Mode). No NRM, a configuração da estação é desbalanceada. Temos uma estação principal e diversas estações secundárias. Uma *estação primária* pode enviar comandos; uma *estação secundária* pode apenas responder a eles. O NRM é usado tanto para enlaces ponto a ponto como multiponto, conforme mostra a Figura 5.20.

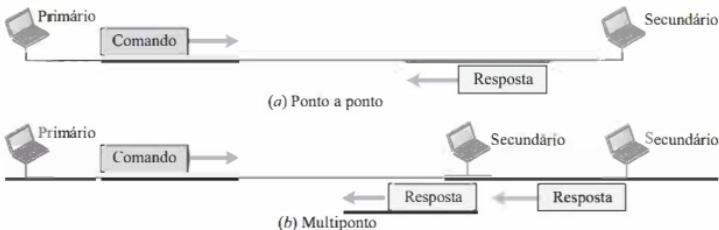



Figura 5.20 Modo de resposta normal.

No modo ABM, a configuração é balanceada. O enlace é ponto a ponto e cada estação pode operar como uma estação primária e secundária (atuando como *peers*), conforme mostra a Figura 5.21. Esse é o modo mais comum atualmente.

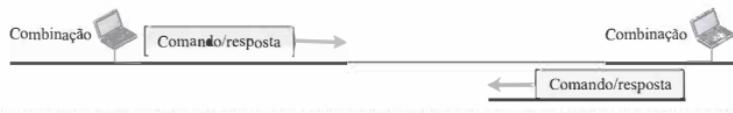



Figura 5.21 Modo balanceado assíncrono.

Quadros

Com o objetivo de fornecer a flexibilidade necessária para suportar todas as opções possíveis nos modos e configurações que acabamos de descrever, o HDLC define três tipos de quadros: *quadros de informação* (quadros-I), *quadros de supervisão* (quadros-S) e *quadros não numerados* (quadros-U, de *unnumbered*). Cada tipo de quadro serve como um envelope para a transmissão de um tipo diferente de mensagem. Os quadros-I são utilizados para transportar dados do usuário e informações de controle relativas aos dados do usuário (mecanismo de carona, ou *piggybacking*). Quadros-S são usados apenas para transportar informações de controle. Quadros-U são reservados para o gerenciamento do sistema. Informações transportadas por quadros-U destinam-se ao gerenciamento do próprio enlace. Cada quadro no HDLC pode conter até seis campos, conforme mostra a Figura 5.22: um campo marcador de início, um campo de endereço, um campo de controle, um campo de informação, um campo de Sequência de Verificação de Quadro (FCS – Frame Check Sequence) e um campo marcador do final do quadro. Nas transmissões de múltiplos quadros, o marcador de fim de um quadro pode servir como o marcador de início do próximo quadro.

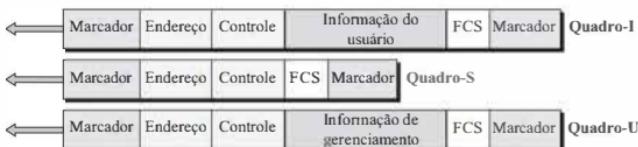



Figura 5.22 Quadros HDLC.

Analisemos os campos e seu uso em diferentes tipos de quadros.

- **Campo marcador.** Contém a sequência de *bits* de sincronização 01111110, que identifica tanto o início como o fim de um quadro.
- **Campo de endereço.** Contém o endereço da estação secundária. Se uma estação principal criou o quadro, esse campo contém o endereço do *destino*. Se uma estação secundária criou o quadro, ele contém o endereço da *origem*. O comprimento do campo de endereço pode ser um *byte* ou vários *bytes*, dependendo das necessidades da rede.
- **Campo de controle.** Consiste em um ou dois *bytes* usados para o controle de fluxo e de erros. A interpretação dos *bits* será discutida mais adiante.
- **Campo de informação.** Contém dados do usuário provenientes da camada de rede ou informação de gerenciamento. O seu comprimento pode variar de uma rede para outra.
- **Campo FCS.** É o campo de detecção de erros do HDLC. Ele contém um CRC de 2 ou de 4 *bytes*.

O campo de controle determina o tipo de quadro e define sua funcionalidade. Portanto, vamos discutir o formato desse campo em detalhes. O formato é específico para o tipo de quadro, conforme mostra a Figura 5.23.

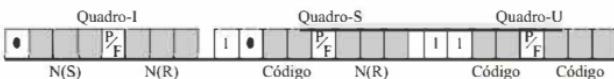



Figura 5.23 Formato do campo de controle para diferentes tipos de quadros.

Campo de controle dos quadros-I Os quadros-I foram projetados para transportar dados do usuário provenientes da camada de rede. Além disso, eles podem incluir informações de controle de fluxo e de erros (mecanismo de carona, ou *piggybacking*). Os subcampos no campo de controle são utilizados para definir essas funções. O primeiro *bit* define o tipo. Se o primeiro *bit* do campo de controle for 0, isto significa que o quadro é um quadro-I. Os próximos 3 *bits*, denotados $N(S)$, definem o número de sequência do quadro. Perceba que, com 3 *bits*, podemos definir um número de sequência entre 0 e 7. Os últimos 3 *bits*, denominados $N(R)$, correspondem ao número de confirmação quando o mecanismo de carona é usado. O único *bit* entre $N(S)$ e $N(R)$ é chamado *bit P/F*. O campo *P/F* consiste em um único *bit* com dupla finalidade. Ele só tem algum significado quando vale 1, podendo significar consulta ou final: significa *consulta* quando o quadro é enviado por uma estação primária para uma secundária (quando o campo de endereço contém o endereço do destinatário) e *final* quando o quadro é enviado de uma estação secundária para uma primária (quando o campo de endereço contém o endereço do remetente).

Campo de controle dos quadros-S Quadros de supervisão são usados para o controle de fluxo e de erros sempre que o mecanismo de carona for impossível de ser usado ou inadequado. Os quadros-S não apresentam campos de informação. Se os primeiros 2 *bits* do campo de controle forem 10, isto significa que o quadro é um quadro-S. Os últimos 3 *bits*, denotados $N(R)$, correspondem ao número de confirmação positiva (ACK) ou negativa (NAK), dependendo do tipo de quadro-S. Os 2 *bits* denominados *código* são usados para definir o tipo do quadro-S em si. Com 2 *bits*, podemos ter quatro tipos de quadros-S, descritos a seguir:

- **Receptor pronto** (RR – Receive Ready). Se o valor do subcampo de código for 00, o quadro-S é do tipo RR, que confirma a recepção correta de um quadro (ou grupo de quadros). Nesse caso, o valor do campo $N(R)$ define o número de confirmação.
- **Receptor não pronto** (RNR – Receive Not Ready). Se o valor do subcampo de código for 10, o quadro-S é do tipo RNR, que corresponde a um quadro RR com funções adicionais. Ele confirma o recebimento de um quadro ou grupo de quadros, mas anuncia que o receptor está ocupado e não pode receber mais quadros. Ele atua como uma espécie de mecanismo de controle de congestionamento, pedindo ao remetente para reduzir a taxa de transmissão. O valor de $N(R)$ define o número de confirmação.
- **Rejeição** (REJ – Reject). Se o valor do subcampo de código for 01, o quadro-S é do tipo REJ, um quadro de NAK, mas não como aquele usado no ARQ da Repetição Seletiva. Trata-se de um NAK que pode ser usado no ARQ *Go-Back-N* para melhorar a eficiência do processo, informando ao remetente, antes que o temporizador do remetente expire, que o último quadro foi perdido ou danificado. O valor do campo $N(R)$ corresponde ao número de confirmação negativa.
- **Rejeição seletiva** (SREJ – Selective Reject). Se o valor do subcampo de código for 11, o quadro-S é um SREJ, um quadro de NAK usado no ARQ de Repetição Seletiva. Perceba

que o protocolo HDLC utiliza o termo *rejeição seletiva*, em vez de *repetição seletiva*. O valor do campo $N(R)$ corresponde ao número da confirmação negativa.

Campo de controle dos quadros-U Quadros não numerados são usados para trocar informações de gerenciamento de sessão e de controle entre dispositivos conectados. Ao contrário dos quadros-S, os quadros-U carregam um campo de informação, mas ele é utilizado para informações do sistema de gerenciamento, não para dados do usuário. Assim como acontece com os quadros-S, no entanto, boa parte das informações transportadas por quadros-U está contida em códigos incluídos no campo de controle. Os códigos dos quadros-U são divididos em duas seções: um prefixo de 2 *bits* antes do *bit* P/F e um sufixo de 3 *bits* após o *bit* P/F. Em conjunto, esses dois segmentos (5 *bits*) podem ser usados para criar até 32 tipos de quadros-U diferentes.

Protocolo Ponto a Ponto

Um dos protocolos mais usados para acesso ponto a ponto é o **Protocolo Ponto a Ponto** (PPP – Point-to-Point Protocol). Atualmente, milhões de usuários da Internet que precisam conectar seus computadores domésticos ao servidor de um ISP utilizam o PPP. A maioria desses usuários possui um modem tradicional; eles se conectam à Internet por meio de uma linha telefônica, a qual fornece os serviços da camada física. Entretanto, para controlar e gerenciar a transferência de dados, é necessário um protocolo ponto a ponto na camada de enlace de dados. O PPP é, de longe, o mais comum.

Serviços

Os projetistas do PPP incluíram diversos serviços para torná-lo adequado para uso como um protocolo ponto a ponto, mas ignoraram alguns serviços tradicionais com o objetivo de simplificá-lo.

Serviços fornecidos pelo PPP O PPP define o formato do quadro a ser trocado entre os dispositivos e também como dois dispositivos podem negociar o estabelecimento da conexão e a troca de dados. O PPP foi projetado para aceitar dados de várias camadas de rede (não apenas IP). A autenticação também é prevista no protocolo, mas é opcional. A nova versão do PPP, denominada *PPP Multienlace*, fornece conexões sobre múltiplos enlaces. Uma característica interessante do PPP é que ele fornece configuração de endereços de rede. Isto é particularmente útil quando um usuário doméstico precisa de um endereço de rede temporário para se conectar à Internet.

Serviços não fornecidos pelo PPP O PPP não fornece controle de fluxo. Um remetente pode enviar vários quadros um após o outro sem se importar com possíveis sobrecargas no receptor. O PPP fornece um mecanismo muito simples para controle de erros. Um campo de CRC é utilizado para detectar erros. Se o quadro estiver corrompido, ele é descartado silenciosamente; o protocolo da camada superior precisa tratar o problema. A ausência de controle de erros e de números de sequência pode fazer com que um pacote seja recebido fora de ordem. O PPP não oferece um mecanismo sofisticado de endereçamento para lidar com quadros em uma configuração multiponto.

Enquadramento

O PPP utiliza quadros orientados a caracteres (ou orientados a *bytes*). A Figura 5.24 mostra o formato de um quadro PPP. A descrição de cada campo é a seguinte:

Marcador	Endereço $(11111111)_2$	Controle $(00000011)_2$	Protocolo	Carga útil	FCS	Marcador
1 *byte*	1 *byte*	1 *byte*	1/2 *bytes*	Variável	2/4 *bytes*	1 *byte*

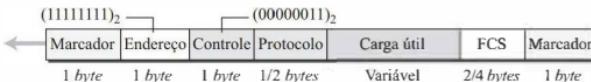



Figura 5.24 Formato do quadro PPP.

- **Marcador.** Um quadro PPP começa e termina com um marcador de 1 *byte* contendo a sequência de *bits* 01111110.
- **Endereço.** O campo de endereço neste protocolo é um valor constante igual a 11111111 (endereço de *broadcast*).
- **Controle.** Campo que tem o valor fixo e constante de 00000011 (imitando os quadros-U do HDLC). Conforme discutiremos mais adiante, o PPP não fornece qualquer controle de fluxo. O controle de erros também é limitado à detecção de erros.
- **Protocolo.** O campo de protocolo define o que está sendo transportado no campo de dados: dados de usuário ou outras informações. Por padrão, esse campo tem 2 *bytes* de comprimento, mas as duas partes podem concordar em usar apenas 1 *byte*.
- **Campo de carga útil (*Payload*).** Carrega os dados do usuário ou outras informações que discutiremos em breve. O campo de dados é uma sequência de *bytes* cujo tamanho máximo por padrão é 1.500 *bytes*; mas esse valor pode ser alterado durante a negociação. Aplica-se o mecanismo de preenchimento de *byte* sobre o campo de dados caso a sequência de *bytes* do marcador de início ou fim apareça nesse campo. Como não há qualquer campo definindo o tamanho do campo de dados, a adição de *bits* de enchimento (*padding*) é necessária se o tamanho for menor do que o valor máximo padrão ou aquele definido via negociação.
- **FCS.** A **Sequência de Verificação de Quadro** (FCS – Frame Check Sequence) é simplesmente um CRC padrão de 2 ou 4 *bytes*.

Preenchimento de *byte* Como o PPP é um protocolo orientado a *bytes*, o marcador no PPP é um *byte* ao qual deve ser adicionada uma sequência de escape sempre que ele aparecer na seção de dados do quadro. O *byte* de escape é 01111101, o que significa que toda vez que uma sequência equivalente a um marcador aparecer nos dados, este *byte* extra é adicionado para informar o receptor de que o próximo *byte* não é um marcador. Obviamente, o *byte* de escape em si deve ser preenchido com outro *byte* de escape.

Transição de fases

Uma conexão PPP passa por fases que podem ser mostradas em um diagrama de *transição de fases* (ver Figura 5.25). O diagrama de transição começa no estado *morto*. Nesse estado, não há qualquer portadora ativa (na camada física) e a linha está silenciosa. Quando um dos dois nós inicia a comunicação, a conexão vai para o estado *estabelecer*. Nele, as opções são negociadas entre as duas partes. Se elas concordam em fazer uma autenticação, o sistema vai para o estado *autenticar*; caso contrário, vai para o estado *rede*. Os pacotes do protocolo de controle de enlace, que será discutido em breve, são usados para essa finalidade. Diversos pacotes podem ser trocados nesse ponto. A transferência de dados acontece no estado *aberto*. Quando uma conexão atinge esse estado, a troca de pacotes de dados pode ser iniciada. A conexão permanece assim até que um dos terminais comunicantes decida encerrar a conexão. Nesse caso, o sistema vai para o estado *encerrar* e permanece nesse estado até que a portadora (sinal da camada física) desapareça, o que novamente leva o sistema para o estado *morto*.

Multiplexação

Embora o PPP seja um protocolo da camada de enlace, ele usa outro conjunto de protocolos para estabelecer a conexão, autenticar as partes envolvidas e transportar os dados da camada de rede. Três conjuntos de protocolos são definidos para tornar o PPP mais poderoso: o Protocolo de Controle de Enlace (LCP – Link Control Protocol), dois Protocolos de Autenticação (APs – Authentication Protocols) e vários Protocolos de Controle de Rede (NCPs – Network Control Protocols). A qualquer momento, um pacote PPP pode transportar dados provenientes de um desses protocolos em seu campo de dados, conforme mostra a Figura 5.26. Observe que há um LCP, dois APs, e vários NCPs. Os dados também podem vir de diversas camadas de rede diferentes.

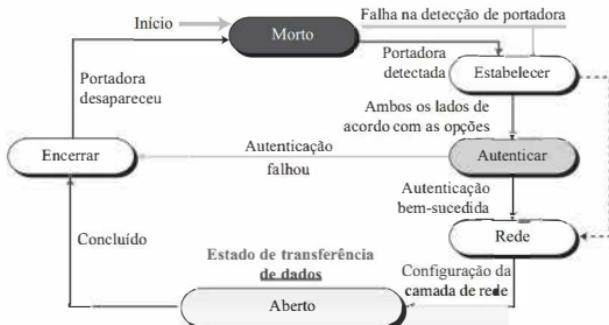



Figura 5.25 Transição de fases.

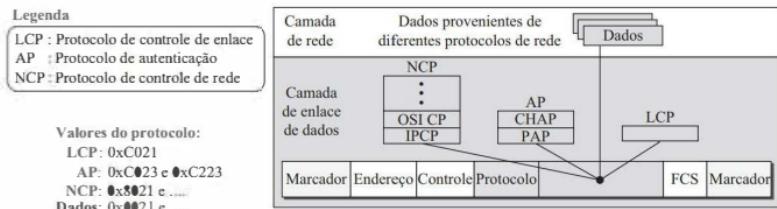



Figura 5.26 Multiplexação no PPP.

Protocolo de Controle de Enlace Este Protocolo (LCP – Link Control Protocol) é responsável por estabelecer, manter, configurar e encerrar conexões. Ele também fornece mecanismos de negociação para definir as opções entre os dois pontos finais. Os pontos finais do enlace devem ambos chegar a um acordo sobre as opções antes que a conexão seja estabelecida.

Protocolos de autenticação A autenticação desempenha um papel muito importante no PPP, pois o protocolo foi projetado para ser utilizado em conexões discadas nas quais a verificação da identidade do usuário é necessária. *Autenticação* significa validar a identidade de um usuário que precisa acessar um conjunto de recursos. O PPP criou dois protocolos de autenticação: o PAP e o CHAP. Note que esses protocolos são utilizados durante a fase de autenticação.

- **PAP.** O *Protocolo de Autenticação por Senha* (PAP – Password Authentication Protocol) é um mecanismo de autenticação simples cujo processo consiste em duas etapas:
  - **a.** O usuário que quer acessar um sistema envia uma identificação de autenticação (normalmente o nome de usuário) e uma senha.
  - **b.** O sistema verifica a validade do par identificação/senha e aceita ou recusa a conexão.
- **CHAP.** *O Protocolo de Autenticação por Desafio-Resposta* (CHAP – Challenge Handshake Authentication Protocol) é um protocolo de autenticação de três vias que proporciona maior segurança do que o PAP. Nesse método, a senha é mantida em sigilo; nunca é enviada pela rede.

a. O sistema envia ao usuário um pacote de desafio contendo o valor de um desafio, que geralmente consiste em alguns *bytes*.

b. O usuário aplica uma função predefinida, que pega o valor do desafio e a senha do próprio usuário e cria um resultado. O usuário envia o resultado para o sistema em um pacote de resposta.

c. O sistema faz o mesmo. Ele aplica a mesma função sobre a senha do usuário (conhecida pelo sistema) e sobre o valor do desafio para criar um resultado; se este for igual ao resultado enviado no pacote de resposta, o acesso é autorizado; caso contrário, o acesso é negado. O CHAP é mais seguro que o PAP, especialmente se o sistema alterar constantemente o valor de desafio. Mesmo que o intruso descubra o valor do desafio e o resultado, a senha continua em segredo.

Protocolos de controle de rede O PPP é um protocolo que aceita múltiplas camadas de rede. Ele pode transportar pacotes de dados da camada de rede provenientes de protocolos definidos pela Internet, OSI, Xerox, DECnet, AppleTalk, Novel, e assim por diante. Para que isto seja possível, o PPP definiu um Protocolo de Controle de Rede (NCP – Network Control Protocol) específico para cada protocolo de rede. Por exemplo, o **Protocolo de Controle de Rede da Internet** (IPCP – Internet Protocol Control Protocol) configura o enlace para que ele transporte pacotes de dados provenientes do IP. O Xerox CP faz o mesmo para os pacotes de dados do protocolo Xerox, e assim por diante. Perceba que nenhum dos pacotes NCP transporta dados da camada de rede; eles apenas configuram o enlace para suportar os dados recebidos dela. O IPCP, protocolo usado para configurar o enlace de modo a transportar pacotes IP na Internet, é de especial interesse para nós.

Dados provenientes da camada de rede Após a configuração da camada de rede ser completada por um dos protocolos NCP, os usuários podem trocar pacotes de dados vindos da camada de rede. Aqui, novamente, há campos de protocolo distintos para diferentes camadas de rede. Por exemplo, se o PPP estiver transportando dados provenientes da camada de rede IP, o valor do campo é $(0021)_{16}$. Se o PPP estiver transportando dados provenientes da camada de rede OSI, o valor do campo de protocolo é $(0023)_{16}$, e assim por diante.

PPP Multienlace

O PPP foi originalmente concebido para uso em um enlace físico ponto a ponto com um único canal. A disponibilidade de múltiplos canais em um mesmo enlace ponto a ponto motivou o desenvolvimento do PPP Multienlace. Nesse caso, um quadro PPP lógico é dividido em vários quadros PPP reais. Um segmento do quadro lógico é transportado como carga útil de um quadro PPP real, conforme mostra a Figura 5.27. Para mostrar que o quadro PPP real está carregando um fragmento de um quadro PPP lógico, o campo de protocolo é preenchido com o valor $(003d)_{16}$. Essa nova funcionalidade introduz maior complexidade. Por exemplo, é preciso adicionar um número de sequência ao quadro PPP real para mostrar a posição de um fragmento no quadro lógico.

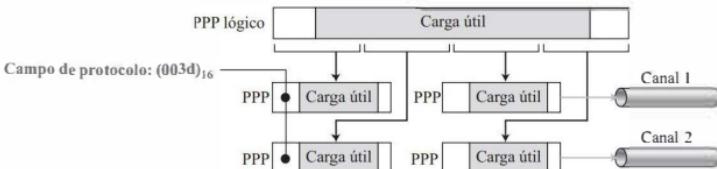



Figura 5.27 PPP Multienlace.

5.3 PROTOCOLOS DE ACESSO MÚLTIPLO

Dissemos que a camada de enlace de dados é dividida em duas subcamadas: Controle de Enlace de Dados (DLC – Data Link Control) e Controle de Acesso ao Meio (MAC – Media Access Control). Discutimos o DLC na seção anterior; nesta seção, falaremos sobre o MAC. Quando estamos usando uma conexão dedicada, como uma linha telefônica discada, precisamos apenas de um protocolo de controle de enlace de dados, tal como o PPP, que gerencia a transferência de dados entre as duas pontas. Por outro lado, se estivermos compartilhando o meio, seja ele fios ou o ar, com outros usuários, precisamos ter um protocolo para primeiramente gerenciar o processo de compartilhamento e então fazer a transferência de dados. Por exemplo, se usarmos um telefone celular para ligar para outro telefone celular, o canal (a banda alocada pela operadora de telefonia) não é dedicado. Uma pessoa a poucos metros de distância pode estar usando a mesma banda para conversar com um amigo.

Quando nós ou estações estão conectados e usando um enlace em comum, denominado enlace multiponto ou de *broadcast*, precisamos de um protocolo de acesso múltiplo para coordenar o acesso a tal enlace. O problema de controlar o acesso ao meio é semelhante às regras de conversação em uma reunião. Os procedimentos garantem que o direito de fala seja concedido a todos, que duas pessoas não falem ao mesmo tempo, que não interrompam umas às outras, que não monopolizem a discussão, e assim por diante. A situação é semelhante nas redes multiponto. Precisamos ter certeza de que cada nó terá acesso ao enlace. O primeiro objetivo é evitar qualquer colisão entre os nós. Se de alguma forma uma colisão ocorrer, o segundo objetivo é resolvê-la.

Muitos protocolos foram concebidos para controlar o acesso a um enlace compartilhado. Podemos categorizá-los em três grupos. Os protocolos pertencentes a cada grupo são mostrados na Figura 5.28.

Figura 5.28 Taxonomia de protocolos de acesso múltiplo discutidos neste capítulo.

5.3.1 Acesso aleatório

Nos métodos de **acesso aleatório** ou **contenção**, nenhuma estação tem preferência ou controle sobre outra estação. A cada instante, uma estação que tem dados para enviar usa um procedimento definido pelo protocolo para tomar a decisão sobre se ela deve ou não enviar os dados. Essa decisão depende do estado do meio (ocioso ou ocupado). Em outras palavras, cada estação pode transmitir quando ela desejar, com a condição de que ela siga o procedimento pré-estabelecido, incluindo uma verificação do estado do meio.

Duas características dão a esse método o seu nome. Em primeiro lugar, não há instantes programados nos quais uma estação pode transmitir. A transmissão é aleatória entre as estações. É por isso que esses métodos são denominados *acesso aleatório*. Em segundo lugar, não há regras específicas sobre qual estação deve se a próxima a enviar dados. As estações competem umas com as outras pelo acesso ao meio. É por essa razão que esses métodos são também chamados métodos de *contenção*.

Em um método de acesso aleatório, cada estação tem direito de acessar o meio sem ser controlada por qualquer outra estação. Entretanto, se mais de uma estação tentar enviar, existe um conflito de acesso – *colisão* – e os quadros serão destruídos ou modificados. Para evitar conflitos de acesso ou para resolvê-los quando eles ocorrerem, cada estação segue um procedimento que responde às seguintes perguntas:

- Quando a estação pode acessar o meio?
- O que a estação pode fazer se o meio estiver ocupado?
- Como a estação pode determinar o sucesso ou a falha na transmissão?
- O que a estação pode fazer se houver um conflito de acesso?

Os métodos de acesso aleatório que estudamos neste capítulo evoluíram a partir de um protocolo muito interessante conhecido como ALOHA, o qual utilizava um procedimento muito simples denominado *Acesso Múltiplo* (MA – Multiple Access). O método foi aperfeiçoado com a adição de um procedimento que força a estação a verificar o estado do meio antes de transmitir. O resultado foi denominado *Acesso Múltiplo com Detecção de Portadora* (CSMA – Carrier Sense Multiple Access). Mais tarde, esse método evoluiu para dois métodos paralelos: *Acesso Múltiplo com Detecção de Portadora/Detecção de Colisão* (CSMA/CD – Carrier Sense Multiple Access/Collision Detection), que diz o que a estação deve fazer quando uma colisão é detectada, e o *Acesso Múltiplo com Detecção de Portadora/Prevenção de Colisão* (CSMA/CA – Carrier Sense Multiple Access/Collision *Avoidance*), que tenta evitar colisões.

ALOHA

O **ALOHA,** o método de acesso aleatório mais antigo, foi desenvolvido na Universidade do Havaí no início de 1970. Ele foi projetado para uso em LANs por rádio (sem fios), mas pode ser utilizado em qualquer meio compartilhado.

É óbvio que há colisões potenciais nesse cenário. O meio (ar) é compartilhado entre as estações. Quando uma estação envia dados, uma outra estação pode tentar fazer o mesmo simultaneamente. Os dados das duas estações colidem e acabam sendo corrompidos.

ALOHA puro

O protocolo ALOHA original é conhecido como **ALOHA puro**. É um protocolo simples, porém elegante. A ideia é que cada estação envie um quadro sempre que tiver um quadro para enviar (acesso múltiplo). No entanto, como existe apenas um canal compartilhado, é possível que ocorram colisões entre os quadros das diferentes estações. A Figura 5.29 mostra um exemplo de colisões de quadros no ALOHA puro.

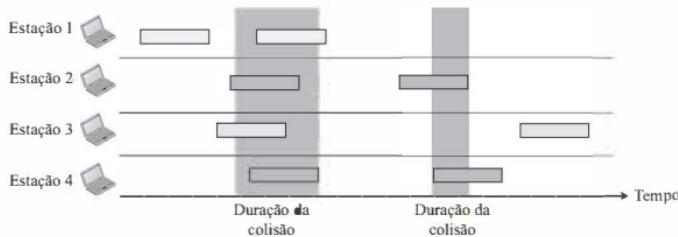



Figura 5.29 Quadros em uma rede usando ALOHA puro.

Existem quatro estações (hipótese pouco realista) que competem umas com as outras pelo acesso ao canal compartilhado. A figura mostra que cada estação envia dois quadros; há, portanto, um total de oito quadros no meio compartilhado. Alguns desses quadros colidem porque vários quadros estão disputando o canal compartilhado. A Figura 5.29 mostra que apenas dois sobrevivem: um quadro da estação 1 e um da estação 3. É importante mencionar que, mesmo que apenas um *bit* de um quadro coexista no canal ao mesmo tempo que um *bit* de outro, uma colisão ocorrerá e ambos os quadros serão destruídos. É óbvio que precisamos reenviar os quadros que foram destruídos durante a transmissão.

O protocolo ALOHA puro depende de confirmações (ACKs) provenientes do receptor. Quando uma estação envia um quadro, ela espera que o receptor envie uma confirmação. Se a confirmação não chegar depois de um tempo-limite, a estação considera que o quadro (ou a confirmação) foi destruído e reenvia o quadro.

Uma colisão pode envolver duas ou mais estações. Se todas essas estações tentarem reenviar os seus quadros logo após o tempo-limite, os quadros colidirão novamente. O ALOHA puro dita que, depois de esgotado o tempo-limite, cada estação espera um intervalo de tempo aleatório antes de reenviar seu quadro. A aleatoriedade ajuda a evitar novas colisões. Denominamos esse intervalo *tempo de retardo* ou *tempo de* back-off, que denotamos por T_B.

O ALOHA puro inclui um segundo método para evitar congestionamentos no canal devido a quadros retransmitidos. Após um número máximo de tentativas de retransmissão K_{max}, uma estação deve desistir e tentar mais tarde. A Figura 5.30 mostra o procedimento usado pelo ALOHA puro com base na estratégia anterior.

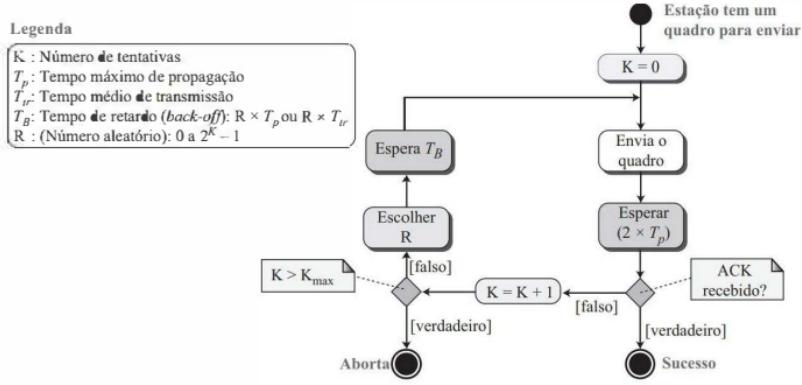



Figura 5.30 Procedimento do protocolo ALOHA puro.

O valor do tempo-limite é igual ao máximo valor possível do atraso de propagação de ida e volta, que é o dobro do tempo necessário para enviar um quadro entre as duas estações que estão separadas pela maior distância ($2 \times T_p$). O tempo de retardo T_B é um valor aleatório que normalmente depende de K (o número de tentativas de transmissão malsucedidas). A fórmula para calcular T_B depende da implementação. Uma fórmula comum é o *retardo exponencial binário*. Nesse método, para cada retransmissão, um multiplicador de $R = 0$ a $2^K - 1$ é escolhido aleatoriamente e multiplicado por T_p (tempo máximo de propagação) ou T_{tr} (o tempo médio necessário para enviar um quadro), resultando no valor de T_B. Perceba que, nesse procedimento, a faixa de números aleatórios aumenta após cada colisão. O valor de K_{max} é geralmente escolhido como 15.

Exemplo 5.11

As estações em uma rede sem fios ALOHA estão separadas por no máximo 600 km de distância. Se considerarmos que os sinais se propagam a uma velocidade de 3×10^8 m/s, verificamos que $T_p = (600 \times 10^3) / (3 \times 10^8) = 2$ ms. Para $K = 2$, os valores que R pode assumir são {0, 1, 2, 3}. Isto significa que T_B pode ser 0, 2, 4 ou 6 ms, dependendo do valor da variável aleatória *R*.

Período vulnerável Calculemos o intervalo de tempo, denominado *período vulnerável*, em que há possibilidade de colisão. Consideramos que as estações enviam quadros de comprimento fixo e que cada quadro leva T_{tr} segundos para ser enviado. A Figura 5.31 mostra o período vulnerável para a estação B.

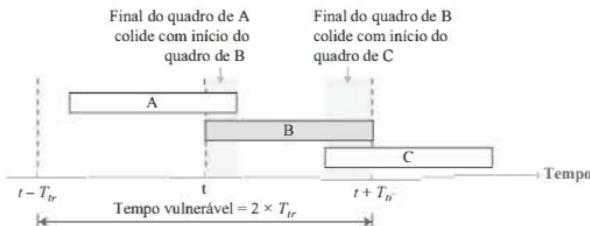



Figura 5.31 Período vulnerável para o protocolo ALOHA puro.

A estação B começa a enviar um quadro no instante *t*. Considere que a estação A começou a enviar seu quadro após o instante $t - T_{tr}$. Isto leva a uma colisão entre os quadros vindos das estações A e B. Além disso, considere que a estação C comece a enviar um quadro antes do instante $t + T_{tr}$. Nesse caso, também ocorre uma colisão entre os quadros das estações B e C.

Observando a Figura 5.31, vemos que o período vulnerável no ALOHA puro, durante o qual uma colisão pode ocorrer, corresponde a duas vezes o intervalo de transmissão de quadros.

$$\textbf{Período vulnerável do ALOHA puro} = 2 \times T_{tr}$$

Exemplo 5.12

Uma rede usando ALOHA puro transmite quadros de 200 *bits* em um canal compartilhado de 200 kbps. Qual é o requisito para que esse quadro seja transmitido livre de colisões?

Solução

O tempo médio de transmissão de quadros T_{tr} é de 200 *bits*/200 kbps ou 1 ms. O período vulnerável vale 2 × 1 ms = 2 ms. Isto significa que nenhuma estação deve começar a enviar quadros 1 ms antes de essa estação iniciar sua transmissão e que nenhuma estação deve começar a enviar quadros durante o período em que essa estação está transmitindo (1 ms).

Vazão Denotamos por *G* o número médio de quadros gerados pelo sistema durante um intervalo de transmissão de quadros. Pode-se, então, provar que o número médio de quadros transmitidos com sucesso pelo ALOHA puro é $V = G \times e^{-2G}$. A vazão máxima V_{max} é 0,184, para $G = 1/2$. (Podemos encontrar esse valor fazendo a derivada de *V* em função de *G* igual a 0, conforme mostrado na seção de Exercícios). Em outras palavras, se meio quadro for gerado em intervalos equivalentes a um intervalo de transmissão de quadros (um quadro para cada dois intervalos de transmissão de

quadros), então 18,4% desses quadros chegarão ao seu destino com sucesso. Espera-se que $G = 1/2$ produza a vazão máxima porque o período vulnerável será duas vezes o intervalo de transmissão de quadros. Portanto, se uma estação gera apenas um quadro nesse período vulnerável (e nenhuma outra estação gera um quadro durante esse intervalo), o quadro vai chegar com sucesso ao seu destino.

A vazão do ALOHA puro é $V = G \times e^{-2G}$.
A vazão máxima é $V_{max} = 1/(2e) = 0{,}184$ quando $G = (1/2)$.

Exemplo 5.13

Uma rede usando ALOHA puro transmite quadros de 200 *bits* em um canal compartilhado de 200 kbps. Qual é a vazão total se o sistema (todas as estações juntas) gera

a. 1.000 quadros por segundo?
b. 500 quadros por segundo?
c. 250 quadros por segundo?

Solução

O intervalo de transmissão de quadros é 200/200 kbps ou 1 ms.

a. Se o sistema gera 1.000 quadros por segundo, ou 1 quadro por milissegundo, então $G = 1$. Nesse caso, $V = G \times e^{-2G} = 0{,}135$ (13,5%). Isto significa que a vazão é de 1.000 × 0,135 = 135 quadros. Apenas 135 quadros a cada 1.000 provavelmente sobreviverão.
b. Se o sistema gera 500 quadros por segundo, ou 1/2 quadro por milissegundo, então $G = 1/2$. Nesse caso, $V = G \times e^{-2G} = 0{,}184$ (18,4%). Isto significa que a vazão é de 500 × 0,184 = 92 e que apenas 92 quadros a cada 500 provavelmente sobreviverão. Note que esse é o caso de vazão *máxima*, em termos percentuais.
c. Se o sistema gera 250 quadros por segundo, ou 1/4 quadro por milissegundo, então $G = 1/4$. Nesse caso, $V = G \times e^{-2G} = 0{,}152$ (15,2%). Isto significa que a vazão é de 250 × 0,152 = 38. Apenas 38 quadros a cada 250 provavelmente sobreviverão.

Slotted ALOHA

O ALOHA puro tem um período vulnerável de $2 \times T_{fr}$. Isto acontece porque não existe uma regra que defina quando a estação pode transmitir. Uma estação pode enviar quadros logo após a outra estação já ter começado a fazê-lo ou logo antes de outra estação ter concluído sua transmissão. O *slotted* ALOHA foi inventado para melhorar a eficiência do ALOHA puro.

No ***slotted* ALOHA**, também conhecido como **ALOHA segmentado** ou **ALOHA discreto,** dividimos o tempo em intervalos de T_{fr} segundos, denominados *faixas* ou *slots*, e forçamos as estações a transmitir apenas no início de cada faixa. A Figura 5.32 mostra um exemplo de colisões de quadros no *slotted* ALOHA.

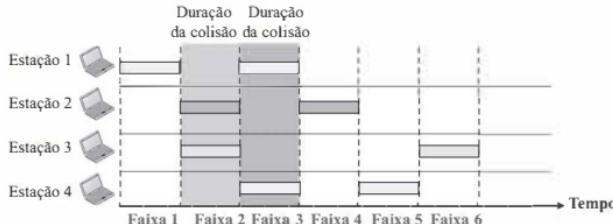



Figura 5.32 Quadros em uma rede baseada em *slotted* ALOHA.

Como as estações podem enviar apenas no início de cada faixa sincronizada de tempo, se uma estação deixa passar esse momento, ela deve esperar até o início da faixa seguinte. Isto significa que a estação que começou a transmissão no início dessa faixa já terá terminado o envio de seu quadro. É claro que ainda existe a possibilidade de colisão se duas estações tentarem transmitir seus quadros no início da mesma faixa. Entretanto, o período vulnerável é agora reduzido pela metade, passando a valer T_{tr}. A Figura 5.33 ilustra essa situação.

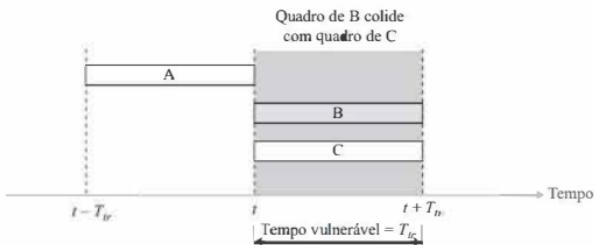



Figura 5.33 Período vulnerável para o protocolo *slotted* ALOHA.

Período vulnerável no *slotted* ALOHA = T_{tr}

Vazão Pode-se provar que o número médio de quadros transmitidos com sucesso pelo *slotted* ALOHA é dado por $V = G \times e^{-G}$. A vazão máxima V_{max} é 0,368, quando $G = 1$. Em outras palavras, se um quadro é gerado em intervalos equivalentes a um intervalo de transmissão de quadros, então 36,8% desses quadros chegarão ao seu destino com sucesso. Espera-se que $G = 1$ produza a vazão máxima porque o período vulnerável será igual ao intervalo de transmissão de quadros. Portanto, se uma estação gerar apenas um quadro nesse período vulnerável (e nenhuma outra estação gerar um quadro durante esse intervalo), o quadro chegará com sucesso ao seu destino.

A vazão para o *slotted* ALOHA é $S = G \times e^{-G}$.
A taxa de transferência máxima é $V_{max} = 0,368$ quando $G = 1$.

Exemplo 5.14

Uma rede baseada em *slotted* ALOHA transmite quadros de 200 *bits* usando um canal compartilhado com uma largura de banda de 200 kbps. Determine a vazão se o sistema (todas as estações juntas) gera

a. 1.000 quadros por segundo.
b. 500 quadros por segundo.
c. 250 quadros por segundo.

Solução

Essa situação é semelhante à do exercício anterior, exceto que a rede usa *slotted* ALOHA em vez do ALOHA puro. O intervalo de transmissão de quadros é 200/200 kbps ou 1 ms.

a. Nesse caso, $G = 1$. Portanto, $V = G \times e^{-G} = 0,368$ (36,8%). Isto significa que a vazão é de 1.000 × 0,368 = 368 quadros. Apenas 368 quadros a cada 1.000 provavelmente sobreviverão. Note que esse é o caso de vazão *máxima*, em termos percentuais.

b. Nesse caso, $G = 1/2$. Portanto, $V = G \times e^{-G} = 0,303$ (30,3%). Isto significa que a vazão é de $500 \times 0,303 = 151$. Apenas 151 quadros a cada 500 provavelmente sobreviverão.

c. Agora $G = 1/4$. Nesse caso, $V = G \times e^{-G} = 0,195$ (19,5%). Isto significa que a vazão é de $250 \times 0,195 = 49$. Apenas 49 quadros a cada 250 provavelmente sobreviverão.

Acesso Múltiplo com Detecção de Portadora

Para minimizar a possibilidade de colisões e, portanto, aumentar a vazão da rede, foi desenvolvido o método CSMA. A probabilidade de colisão pode ser reduzida se uma estação verificar o estado do meio antes de tentar usá-lo. O **Acesso Múltiplo com Detecção de Portadora** (CSMA – Carrier Sense Multiple Access) exige que cada estação ouça o meio (ou verifique o estado do meio) antes de enviar quadros. Em outras palavras, o CSMA é baseado no princípio de "verifique antes de transmitir" ou "ouça antes de falar".

O CSMA pode reduzir a possibilidade de colisão, mas não é capaz de eliminá-la. A razão para isso é mostrada na Figura 5.34, que mostra um modelo espaço-temporal de uma rede CSMA. As estações estão conectadas por meio de um canal compartilhado (geralmente um meio dedicado).

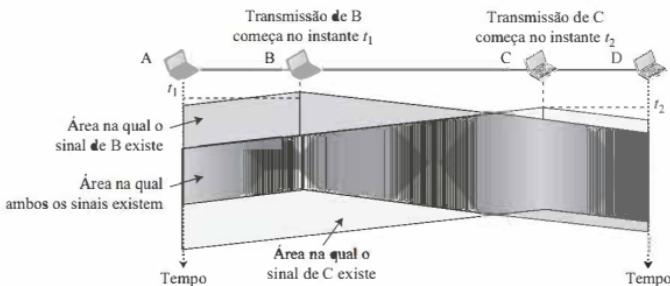



Figura 5.34 Modelo de espaço-temporal de uma colisão no CSMA.

A possibilidade de colisão ainda existe por causa do atraso de propagação; quando uma estação envia um quadro, ainda leva algum tempo (embora bastante curto) para que o primeiro *bit* chegue a cada estação e para que cada estação o perceba. Em outras palavras, uma estação pode verificar o meio e encontrá-lo livre apenas porque o primeiro *bit* enviado por outra estação ainda não foi recebido.

No instante t_1, a estação B verifica o meio e o encontra livre, e por isso envia um quadro. No instante t_2 ($t_2 > t_1$), a estação C verifica o meio e o encontra livre porque, nesse momento, os primeiros *bits* da estação B ainda não chegaram à estação C. A estação C também envia um quadro. Os dois sinais colidem e ambos os quadros são destruídos.

Período vulnerável

O período vulnerável do CSMA equivale ao tempo de propagação T_p. Esse é o intervalo necessário para que um sinal se propague de uma extremidade do meio até a outra. Quando uma estação envia um quadro e qualquer outra estação tenta enviar um quadro durante esse período, uma colisão ocorrerá. Porém, se o primeiro *bit* do quadro atingir a extremidade do meio, todas as estações já terão ouvido o *bit* e não enviarão dados. A Figura 5.35 mostra o pior caso. A estação mais à esquerda, A, envia um quadro no instante t_1, que alcança a estação D, mais à direita, no instante $t_1 + T_p$. A área cinzenta mostra a área vulnerável no tempo e espaço.

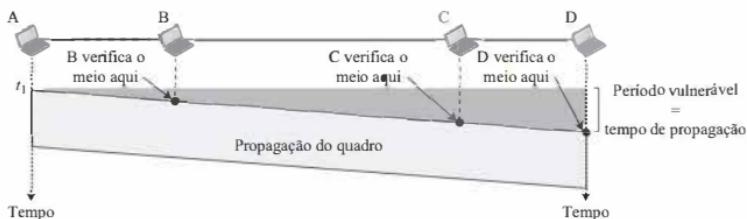



Figura 5.35 Período vulnerável no CSMA.

Métodos de persistência

O que uma estação deve fazer se o canal estiver ocupado? O que uma estação deve fazer se o canal estiver ocioso? Três métodos foram desenvolvidos para responder a essas perguntas: o **método 1-persistente**, o **método não persistente** e o **método *p*-persistente**. A Figura 5.36 mostra o comportamento desses três métodos de persistência quando uma estação encontra um canal ocupado.

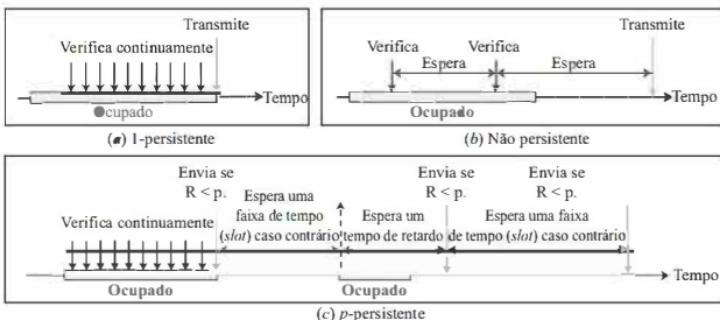



Figura 5.36 Comportamento de três métodos de persistência.

A Figura 5.37 mostra os diagramas de fluxo para esses métodos.

1-persistente O *método 1-persistente* é simples e direto. Nele, após a estação perceber que a linha está ociosa, ela envia o seu quadro imediatamente (com probabilidade 1). Esse método tem a maior chance de colisão porque duas ou mais estações podem perceber a linha ociosa e enviar seus quadros imediatamente. Veremos mais adiante que a Ethernet usa esse método.

Não persistente No *método não persistente*, uma estação que tem um quadro para enviar verifica a linha. Se ela estiver ociosa, a estação envia o quadro imediatamente. Caso não esteja, a estação espera um período aleatório de tempo e então verifica a linha novamente. A abordagem não persistente reduz a possibilidade de colisão, pois é improvável que duas ou mais estações aguardem a mesma quantidade de tempo e tentem o reenvio simultaneamente. No entanto, esse método reduz a eficiência da rede porque o meio permanece ocioso enquanto pode haver estações com quadros para enviar.

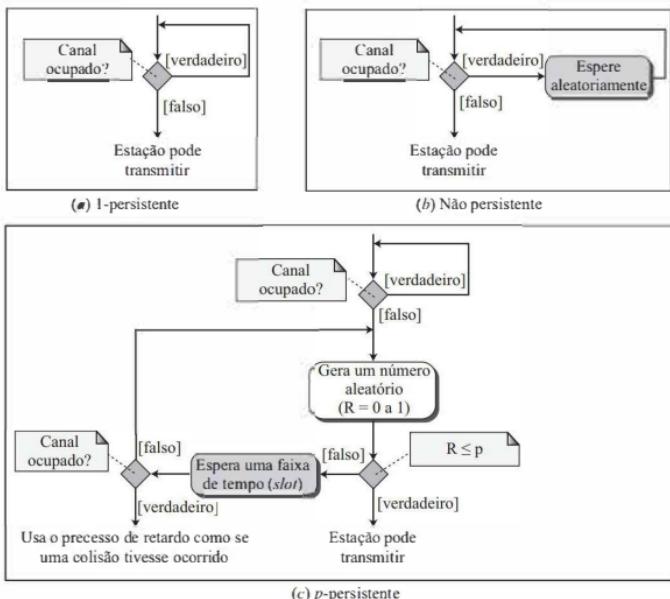



Figura 5.37 Diagrama de fluxo para três métodos de persistência.

p-persistente O método p-*persistente* é utilizado se o canal empregar faixas de tempo (*slots*) com uma duração maior ou igual ao intervalo máximo de propagação. A abordagem *p*-persistente combina as vantagens das duas outras estratégias. Ela reduz a probabilidade de colisão e melhora a eficiência da rede. Nesse método, depois que a estação percebe a linha ociosa, ela segue os seguintes passos:

1. Com probabilidade *p*, a estação envia seu quadro.
2. Com probabilidade $q = 1 - p$, a estação aguarda o início da próxima faixa de tempo e verifica a linha novamente.
   a. Se a linha (meio) estiver ociosa, a estação volta ao passo 1.
   b. Se a linha (meio) estiver ocupada, a estação age como se uma colisão tivesse ocorrido e usa o procedimento de retardo.

CSMA/CD

O método CSMA não especifica o procedimento a ser seguido logo após uma colisão. O **Acesso Múltiplo com Detecção de Portadora/Detecção de Colisão** (CSMA/CD – Carrier Sense Multiple Access/Collision Detection) expande o algoritmo para tratar colisões.

Nesse método, uma estação monitora o meio após enviar um quadro para verificar se a transmissão foi bem-sucedida. Se assim for, o trabalho da estação está terminado. Se, no entanto, houver uma colisão, o quadro é enviado novamente.

Para entender melhor o CSMA/CD, observaremos os primeiros *bits* transmitidos pelas duas estações envolvidas na colisão. Embora cada estação continue a enviar os *bits* do quadro até detectar a colisão, mostraremos o que acontece quando os primeiros *bits* colidem. Na Figura 5.38, as estações A e C estão envolvidas na colisão.

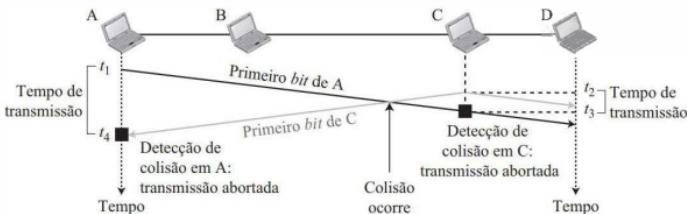



Figura 5.38 Colisão dos primeiros *bits* no CSMA/CD.

No instante t_1, a estação A executa o seu procedimento de persistência e começa a enviar os *bits* do seu quadro. No instante t_2, a estação C ainda não detectou o primeiro *bit* enviado pela estação A. A estação C executa o seu procedimento de persistência e começa a enviar os *bits* de seu quadro, que se propagam tanto para a esquerda como para a direita. A colisão ocorre algum tempo após o instante t_2. A estação C detecta a colisão no instante t_3, quando recebe o primeiro *bit* do quadro de A. A estação C imediatamente (ou após um curto período de tempo, mas aqui consideramos imediatamente) interrompe a transmissão. A estação A detecta colisão no instante t_4, quando recebe o primeiro *bit* do quadro de C; ela também interrompe a transmissão imediatamente. Observando a figura, vemos que A transmite por um intervalo de tempo igual a $t_4 - t_1$, enquanto a transmissão de C dura $t_3 - t_2$.

Agora que conhecemos a duração das duas transmissões, podemos mostrar um gráfico mais completo na Figura 5.39.

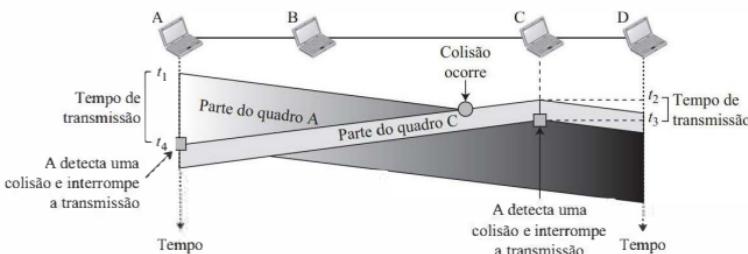



Figura 5.39 Colisão e interrupção da transmissão no CSMA/CD.

Tamanho mínimo dos quadros

Para que o CSMA/CD funcione, precisamos impor uma restrição ao tamanho dos quadros. Antes de enviar o último *bit* do quadro, a estação emissora deve detectar uma colisão, caso ela ocorra, e abortar a transmissão. Isto acontece porque a estação, após todo o quadro ter sido enviado, não mantém uma

cópia dele e não monitora a linha para detectar colisões. Portanto, o intervalo de transmissão de quadros T_{tr} deve ser pelo menos duas vezes o valor máximo do tempo de propagação T_p. Para entender essa razão, consideremos o cenário de pior caso. Se as duas estações envolvidas em uma colisão estiverem separadas pela distância máxima da rede, o sinal da primeira leva um tempo T_p para alcançar a segunda, e o efeito da colisão toma outro intervalo de tempo T_p para alcançar a primeira. Desse modo, exige-se que a primeira estação ainda esteja transmitindo após um intervalo de tempo de $2T_p$.

Exemplo 5.15

Uma rede usando CSMA/CD apresenta uma largura de banda de 10 Mbps. Se o tempo máximo de propagação (incluindo os atrasos observados nos dispositivos e ignorando o tempo necessário para enviar um sinal de interferência, conforme veremos mais adiante) é de 25,6 μs, qual é o tamanho mínimo do quadro?

Solução

O tempo mínimo de transmissão de quadros é $T_{fr} = 2 \times T_p = 51,2$ μs. Isto significa que, no pior caso, uma estação precisa transmitir por um período de 51,2 μs para detectar a colisão. O tamanho mínimo do quadro é de 10 Mbps × 51,2 μs = 512 *bits* ou 64 *bytes*. É realmente o tamanho mínimo do quadro para a Ethernet Padrão, conforme veremos mais adiante.

Procedimento

Agora, observemos o diagrama de fluxo do CSMA/CD na Figura 5.40. Ele é semelhante ao usado para o protocolo ALOHA, mas existem diferenças.

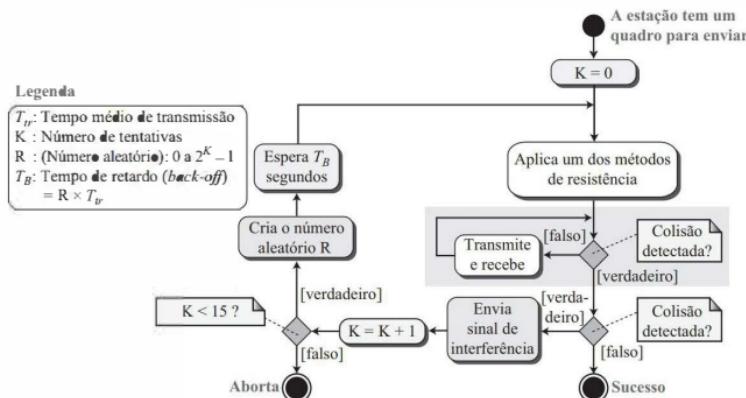



Figura 5.40 Diagrama de fluxo para o CSMA/CD.

A primeira diferença é a inserção do processo de persistência. Precisamos verificar o canal antes de começar a enviar o quadro usando um dos processos de persistência discutidos anteriormente (não persistente, 1-persistente ou *p*-persistente). A caixa correspondente pode ser substituída por um dos processos de persistência mostrados na Figura 5.37.

A segunda diferença refere-se à transmissão de quadros. No ALOHA, primeiro transmitimos o quadro inteiro e depois esperamos por uma confirmação. No CSMA/CD, a transmissão e a detecção de colisão fazem parte de um processo contínuo. Não enviamos o quadro inteiro e

depois verificamos se houve uma colisão. A estação transmite e recebe contínua e simultaneamente (usando duas portas diferentes ou uma porta bidirecional). Usamos um laço para mostrar que a transmissão é um processo contínuo. Monitoramos constantemente a fim de detectar uma dentre duas condições: ou a transmissão foi concluída ou uma colisão foi detectada. Qualquer um desses eventos interrompe a transmissão. Quando saímos do laço, se uma colisão não tiver sido detectada, significa que a transmissão foi concluída com sucesso, ou seja, o quadro todo foi transmitido. Caso contrário, ocorreu uma colisão.

A terceira diferença é o envio de um curto *sinal de interferência* (*jamming*) para se certificar de que todas as outras estações tomaram ciência da colisão.

Nível de energia

Podemos dizer que o nível de energia em um canal pode assumir três valores: zero, normal e anormal. No nível zero, o canal está ocioso. No nível normal, uma estação capturou com sucesso o canal e está enviando seu quadro. No nível anormal, há uma colisão e o nível da energia é duas vezes o nível normal. Uma estação que tem um quadro para enviar ou está enviando um quadro precisa monitorar o nível de energia para determinar se o canal está ocioso, ocupado ou no modo de colisão. A Figura 5.41 mostra essa situação.

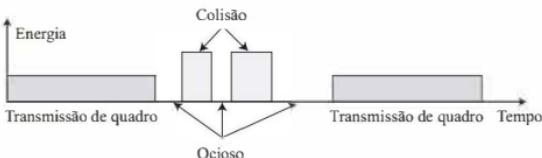



Figura 5.41 Nível de energia durante a transmissão, ociosidade ou colisão.

Vazão

A vazão do CSMA/CD é superior à do ALOHA puro e do *slotted* ALOHA. A vazão máxima é obtida com um valor diferente de G e depende do método de persistência e do valor de p na abordagem p-persistente. Para o método 1-persistente, a taxa de transferência máxima é de cerca de 50% quando $G = 1$. Para o método não persistente, a vazão máxima pode chegar a 90% quando G estiver entre 3 e 8.

Ethernet tradicional

Um dos protocolos de LAN que utilizava o CSMA/CD é a Ethernet tradicional com uma taxa de transferência de dados de 10 Mbps. Discutiremos as LANs Ethernet no final do capítulo, mas é bom saber que a Ethernet tradicional era uma LAN de *broadcast* que utilizava o método 1-persistente para o controle de acesso ao meio de comunicação compartilhado. As versões posteriores da Ethernet passaram a usar métodos de acesso diferentes do CSMA/CD pelas razões que discutiremos na próxima seção, na qual tratamos de LANs com fios.

CSMA/CA

Uma variante do método CSMA é o **Acesso Múltiplo com Detecção de Portadora/Prevenção de Colisão** (CSMA/CA – Carrier Sense Multiple Access/Collision Avoidance), que é usado em LANs sem fios. Adiaremos a discussão sobre esse método para o Capítulo 6.

5.3.2 Acesso controlado

No **acesso controlado**, as estações consultam umas às outras para determinar qual estação tem o direito de enviar quadros. A estação não pode enviar a menos que ela tenha sido autorizada por outras estações. Discutimos três métodos de acesso controlado.

Reserva

No método de **reserva**, uma estação precisa fazer uma reserva antes de enviar dados. O tempo é dividido em intervalos discretos; em cada um, um quadro de reserva precede os quadros de dados enviados naquele intervalo.

Se existem *N* estações no sistema, há exatamente *N* mini-intervalos de reserva em um quadro de reserva. Cada mini-intervalo pertence a uma estação. Quando uma estação precisa enviar um quadro de dados, ela faz uma reserva no seu próprio mini-intervalo. As estações que fizeram reservas podem enviar seus quadros de dados após o quadro de reserva.

A Figura 5.42 mostra uma situação com cinco estações e um quadro de reserva contendo cinco mini-intervalos. No primeiro intervalo, apenas as estações 1, 3 e 4 fizeram reservas. No segundo, apenas uma estação fez uma reserva.

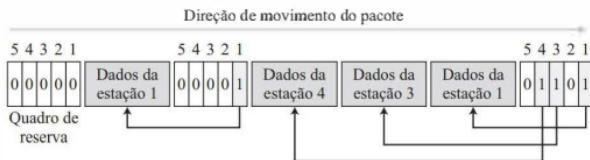



Figura 5.42 Método de acesso por reserva.

Varredura

O método de **varredura** (também conhecido como *polling*) funciona em topologias nas quais um dispositivo é designado como *estação primária* e os outros dispositivos são designados *estações secundárias*. Todas as trocas de dados devem ser feitas por meio do dispositivo primário, mesmo quando o destino final for um dispositivo secundário. O dispositivo primário controla o enlace; os secundários seguem as suas instruções. É tarefa do dispositivo primário determinar qual dispositivo está autorizado a utilizar o canal em um dado instante. O dispositivo primário, portanto, é sempre o iniciador de uma sessão (ver Figura 5.43). Esse método usa funções de varredura e seleção para evitar colisões. No entanto, a desvantagem é que, se a estação primária falhar, o sistema todo falha.

Seleção

A função de seleção é usada sempre que o dispositivo primário tem algo para enviar. Lembre-se de que a estação primária controla o enlace. Se ela não estiver enviando nem recebendo dados, ela sabe que o enlace está disponível. Se a estação primária tiver algo para enviar, ela envia. O que ela não sabe, no entanto, é se o dispositivo de destino está preparado para receber. Assim, a estação primária deve alertar a secundária sobre a transmissão que virá e deve aguardar uma confirmação (ACK) de que a estação secundária está pronta para receber dados. Antes de enviar os dados, a estação primária cria e transmite um quadro de seleção (SEL), que tem um campo no qual está o endereço da estação secundária de destino.

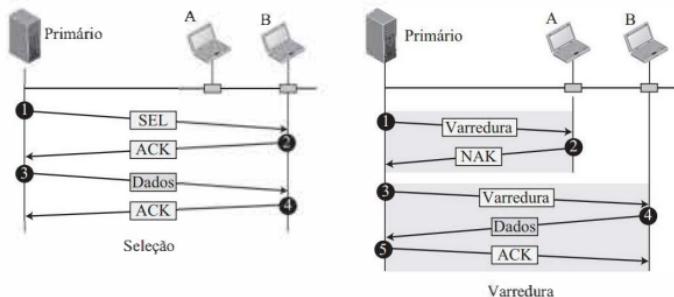



Figura 5.43 Funções de seleção e varredura no método de acesso por varredura.

Varredura

A função de varredura (*pool*) é utilizada pelo dispositivo primário para solicitar transmissões dos dispositivos secundários. Quando a estação primária está pronta para receber dados, ela deve perguntar a cada dispositivo, um de cada vez, se ele tem algo para enviar. Quando o primeiro dispositivo secundário é abordado, ele responde com um quadro de NAK caso não tenha algo para enviar, ou com dados (na forma de um quadro de dados) caso contrário. Se a resposta for negativa (um quadro NAK), a estação primária pergunta à próxima estação secundária como anteriormente, até encontrar uma estação com dados para enviar. Quando a resposta for positiva (um quadro de dados), a estação primária lê o quadro e retorna uma confirmação (quadro ACK), verificando sua recepção.

Passagem de ficha

No método de **passagem de ficha**, as estações da rede são organizadas em um anel lógico. Em outras palavras, para cada estação existe um *predecessor* e um *sucessor*. O predecessor é a estação logicamente anterior à estação em questão no anel, enquanto o sucessor é a estação seguinte à estação no anel. A estação atual é aquela que está acessando o canal no momento. O direito de realizar esse acesso foi passado pelo antecessor da estação atual. Esse direito será transferido ao sucessor da estação atual quando ela não tiver mais dados para enviar.

Mas como o direito de acessar o canal é passado de uma estação para outra? Nesse método, um pacote especial denominado *ficha* (*token*) circula através do anel. A posse da ficha dá à estação o direito de acessar o canal e enviar seus dados. Quando uma estação tem alguns dados para enviar, ela espera até receber a ficha de seu antecessor. Em seguida, detém a ficha e envia seus dados. Quando a estação não tiver mais dados para enviar, ela libera a ficha, passando-a para a próxima estação no anel lógico. A estação não pode enviar dados até receber a ficha novamente na próxima rodada. Nesse processo, quando uma estação recebe a ficha e não tem dados para enviar, ela simplesmente passa a ficha para a estação seguinte.

O gerenciamento de fichas é necessário nesse método de acesso. Deve haver um limite de tempo durante o qual as estações podem manter a posse da ficha, que deve ser monitorada para garantir que não seja perdida ou destruída. Por exemplo, se a estação que estiver de posse da ficha falhar, a ficha desaparecerá da rede. Outra função do gerenciamento de fichas é atribuir prioridades às estações e aos tipos de dados a serem transmitidos. Finalmente, o gerenciamento de fichas é necessário para garantir que estações com baixa prioridade liberarem a ficha para estações de prioridade mais alta.

Anel lógico

Em uma rede que usa o método de passagem de ficha, as estações não precisam estar fisicamente conectadas em um anel; ele pode ser lógico. A Figura 5.44 mostra quatro topologias físicas diferentes que podem criar um anel lógico.

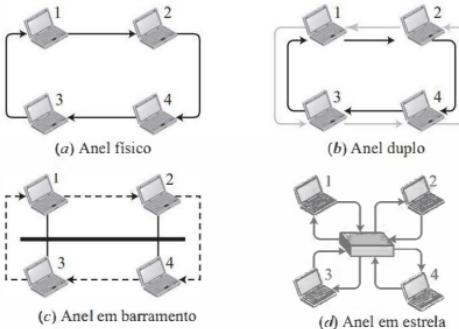



Figura 5.44 Anel lógico e topologia física no método de acesso por passagem de ficha.

Na topologia em anel físico, quando uma estação envia a ficha para o seu sucessor, ela não pode ser vista pelas outras estações; o sucessor é a próxima estação na linha. Isto significa que a ficha não precisa carregar o endereço do próximo sucessor. O problema com essa topologia é que, se um dos elos – o meio entre duas estações adjacentes – falhar, o sistema todo falha.

A topologia em anel duplo utiliza um segundo anel (auxiliar) que opera na direção inversa à do anel principal. O segundo anel é usado apenas para emergências (análogo a um pneu sobressalente em um carro). Se um dos elos do anel principal falhar, o sistema automaticamente combina os dois anéis para formar um anel temporário. Após o elo falho ser restabelecido, o anel auxiliar volta a ficar ocioso. Perceba que, para essa topologia funcionar, cada estação precisa ter duas portas de transmissão e duas portas de recepção. As redes Token Ring de alta velocidade conhecidas como Interface de Dados Distribuídos por Fibra (FDDI – Fiber Distributed Data Interface) e Interface de Dados Distribuídos por Cobre (CDDI – Copper Distributed Data Interface) adotam essa topologia.

Na topologia de anel em barramento (*bus ring*), também chamada de ficha em barramento (*bus token*), as estações estão conectadas a um único cabo denominado barramento. Elas formam, entretanto, um anel lógico, pois cada estação conhece o endereço de seu sucessor (e também do predecessor para fins de gerenciamento de fichas). Quando uma estação termina de enviar seus dados, ela libera a ficha e insere o endereço de seu sucessor na ficha. Apenas a estação cujo endereço corresponde ao endereço de destino da ficha fica com a ficha de acesso ao meio compartilhado. A LAN conhecida como Token Bus, padronizada pelo IEEE, adota esta topologia.

Na topologia de anel em estrela, a topologia física é uma estrela. Existe um *hub*, no entanto, que atua como conector da rede. O cabeamento dentro do *hub* é o que cria o anel; as estações são conectadas a ele por meio de duas conexões com fios. Essa topologia torna a rede menos propensa a falhas, porque se um elo falhar, ele pode ser ignorado pelo *hub* e as estações restantes podem continuar a operar normalmente. A adição e remoção de estações no anel também são facilitadas. Essa topologia ainda é usada na LAN Token Ring concebida pela IBM.

5.3.3 Canalização

A **canalização** (ou *divisão de canal*, como às vezes é denominada) é um método de acesso múltiplo em que a largura de banda disponível em um enlace é compartilhada no tempo, na frequência ou por meio de códigos, entre diferentes estações. Como tais métodos são normalmente usados em redes sem fios, deixamos a discussão sobre eles para o próximo capítulo.

5.4 ENDEREÇAMENTO NA CAMADA DE ENLACE

A próxima questão que precisamos discutir com relação à camada de enlace de dados são os endereços da camada de enlace. No Capítulo 4, discutimos endereços IP como os identificadores na camada de rede que definem os pontos exatos na Internet aos quais as estações de origem e de destino estão conectadas. No entanto, em uma internet não orientada à conexão como a Internet, não podemos fazer um datagrama chegar ao seu destino utilizando apenas endereços IP. A razão é que cada datagrama na Internet, mesmo que venha de uma mesma estação de origem e vá para uma mesma estação de destino, pode tomar um caminho diferente entre elas. Os endereços IP de origem e de destino definem as duas extremidades da comunicação, mas não podem definir os enlaces pelos quais o datagrama deve passar.

É preciso ter em mente que os endereços IP de um datagrama não devem ser modificados. Se o endereço IP de destino em um datagrama for alterado, o pacote nunca chegará ao seu destino; se o de origem for alterado, a estação de destino e os roteadores não serão capazes de se comunicar com a origem caso uma resposta precise ser retornada ou caso seja necessário lhes reportar um erro (ver a discussão sobre ICMP no Capítulo 4).

A discussão anterior mostra que precisamos de outro mecanismo de endereçamento em internets não orientadas à conexão: os endereços da camada de enlace dos dois nós. Um *endereço da camada de enlace* é também conhecido como *endereço do enlace*, *endereço físico* ou *endereço MAC*. Neste livro, usamos esses termos como sinônimos.

Como um enlace é controlado na camada de enlace de dados, os endereços precisam pertencer à camada de enlace de dados. Quando um datagrama passa da camada de rede para a camada de enlace de dados, o datagrama é encapsulado em um quadro e dois endereços da camada de enlace de dados são adicionados ao cabeçalho do quadro. Eles são alterados cada vez que o quadro passa de um enlace para o outro. A Figura 5.45 ilustra esse conceito em uma internet de pequeno porte.

Na internet da Figura 5.45, temos três enlaces e dois roteadores. Mostramos também apenas duas estações: Alice (origem) e Bob (destino). Para cada estação, mostramos dois endereços, os endereços IP (I) e os endereços da camada de enlace (E). Observe que um roteador tem tantos pares de endereços quanto o número de enlaces aos quais o roteador está conectado. Mostramos três quadros, um em cada enlace. Cada quadro carrega os mesmos endereços IP de origem e destino (**I_1** e **I_8**), mas os endereços da camada de enlace no quadro mudam de enlace para enlace. No enlace 1, os endereços da camada de enlace são E_1 e E_2. No enlace 2, eles são E_4 e E_5. No 3, eles são E_7 e E_8. Perceba que os endereços IP e os endereços da camada de enlace não estão na mesma ordem. Para endereços IP, o endereço de origem vem antes do endereço de destino; para endereços da camada de enlace, o endereço de destino vem antes do de origem. Os datagramas e os quadros foram projetados assim e seguimos esse padrão. Podemos levantar várias questões:

- Se o endereço IP de um roteador não aparece em qualquer datagrama enviado de uma origem para um destino, por que precisamos atribuir endereços IP para os roteadores? A resposta é que, em alguns protocolos, um roteador pode atuar como o emissor ou receptor de um datagrama. Por exemplo, nos protocolos de roteamento que discutimos no Capítulo 4, um roteador é um emissor ou um receptor de uma mensagem. As comunicações nesses protocolos se dão entre os roteadores.

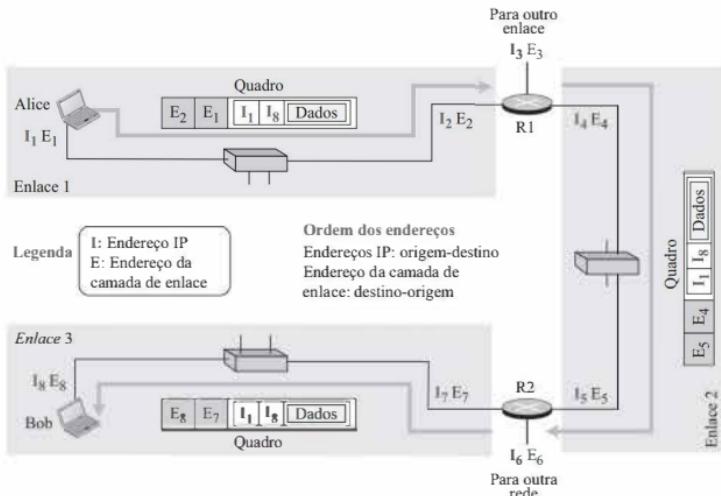



Figura 5.45 Endereços IP e da camada de enlace em uma pequena internet.

- Por que precisamos de mais de um endereço IP em um roteador, um para cada interface? A resposta é que uma interface é uma conexão de um roteador em um enlace. Dissemos que um endereço IP define um ponto na Internet ao qual um dispositivo está conectado. Um roteador com *n* interfaces está conectado à Internet em *n* pontos. É o caso de uma casa que fique na esquina de uma rua e que tenha duas portas; cada porta tem o endereço relativo à rua correspondente.
- Como os endereços IP de origem e destino de um pacote são determinados? A resposta é que a estação deve saber o seu próprio endereço IP, que é usado como endereço IP de origem no pacote. Conforme discutimos no Capítulo 2, a camada de aplicação utiliza os serviços do DNS para localizar o endereço de destino do pacote e o passa para a camada de rede para que o mesmo seja inserido no pacote.
- Como os endereços da camada de enlace de origem e de destino são determinados para cada enlace? Mais uma vez, cada salto (roteador ou estação) deve saber o seu próprio endereço da camada de enlace, conforme discutiremos mais adiante neste capítulo. O endereço de destino da camada de enlace é determinado por meio do Protocolo de Resolução de Endereços (ARP – Address Resolution Protocol), que abordaremos em breve.
- Qual é o tamanho dos endereços da camada de enlace? A resposta é que isto depende do protocolo utilizado pelo enlace. Embora tenhamos apenas o protocolo IP para a Internet toda, podemos usar diferentes protocolos da camada de enlace de dados em enlaces distintos. Isto significa que podemos definir o tamanho do endereço quando discutimos os diferentes protocolos da camada de enlace, o que será feito para redes com fios neste capítulo e para redes sem fio no próximo.

Exemplo 5.16

Como discutiremos mais adiante neste capítulo, os endereços da camada de enlace no tipo mais comum de LAN, a Ethernet, apresentam 48 *bits* (seis *bytes*) que são representados como 12 dígitos hexadecimais separados por dois pontos; por exemplo, o seguinte é um endereço da camada de enlace de um computador.

A2:34:45:11:92:F1

Protocolo de Resolução de Endereços

Sempre que um nó tem um datagrama IP para enviar para outro nó em um enlace, ele tem o endereço IP do nó de destino. A estação de origem sabe o endereço IP do roteador-padrão. Cada roteador, exceto pelo último no caminho, obtém o endereço IP do próximo roteador usando sua tabela de roteamento (na coluna próximo salto). O último roteador conhece o endereço IP da estação de destino. No entanto, o endereço IP do próximo nó não é útil para movimentar um quadro por meio de um enlace; precisamos do endereço da camada de enlace do nó seguinte. É nesse momento que o **Protocolo de Resolução de Endereços** (ARP – Address Resolution Protocol) mostra-se útil. O protocolo ARP é um dos protocolos auxiliares definidos na camada de rede, conforme mostrado na Figura 5.46. Ele pertence à camada de rede, mas adiamos a sua discussão até agora porque ele mapeia um endereço IP para um endereço de enlace lógico. O ARP recebe um endereço IP vindo do protocolo IP, mapeia o endereço para o endereço correspondente da camada de enlace, e passa o resultado para a camada de enlace de dados.

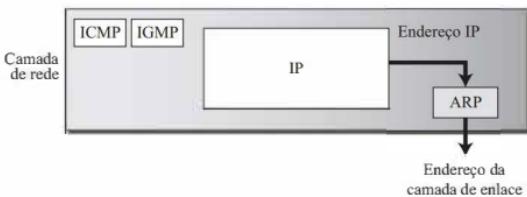



Figura 5.46 Posição do ARP na pilha de protocolos TCP/IP.

Sempre que uma estação ou um roteador precisa encontrar o endereço da camada de enlace de outra estação ou outro roteador em sua rede, ele envia um pacote de pedido ARP. O pacote inclui os endereços da camada de enlace e IP do remetente e o endereço IP do destinatário. Como o remetente não sabe o endereço da camada de enlace do destinatário, a consulta é transmitida através do enlace com o endereço de *broadcast* da camada de enlace, que discutiremos mais adiante para cada protocolo (ver Figura 5.47).

Todas as estações e roteadores na rede recebem e processam o pacote de pedido ARP, mas somente o destinatário correto reconhece seu endereço IP e envia de volta um pacote de resposta ARP. O pacote de resposta contém os endereços IP e da camada de enlace do destinatário. O pacote é enviado via *unicast*, diretamente para o nó que enviou o pacote de pedido.

Na Figura 5.47*a*, o sistema do lado esquerdo (A) tem um pacote que precisa ser entregue a outro sistema (B) cujo endereço IP é I_2. O sistema A precisa entregar o pacote para sua camada de enlace de dados para a entrega de fato, mas ele não sabe o endereço físico do destinatário. Ele usa os serviços do ARP, pedindo ao protocolo que envie um pacote de pedido ARP via *broadcast*, no qual seja solicitado o endereço físico de um sistema cujo endereço IP é I_2.

Esse pacote é recebido por todos os sistemas na rede física, mas apenas o sistema B responderá, conforme mostra a Figura 5.47*b*. O sistema B envia um pacote de resposta ARP que inclui seu endereço físico. Agora, o sistema A pode enviar todos os pacotes que ele tem para esse destino usando o endereço físico recebido.

(*a*) Pedido ARP é enviado via *broadcast*

(*b*) Resposta ARP é enviada via *unicast*

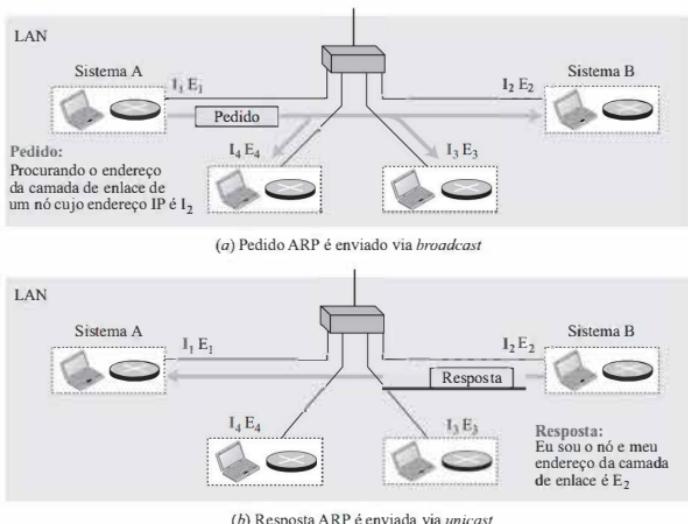



Figura 5.47 Operação do ARP.

Formato dos pacotes

A Figura 5.48 mostra o formato de um pacote ARP. Os nomes dos campos são autoexplicativos. O campo *tipo de hardware* define o tipo do protocolo da camada de enlace; para a Ethernet,

0	8	16	31
Tipo de *hardware*		Tipo de protocolo	
Comprimento do *hardware*	Comprimento do protocolo	Operação **Pedido:1, Resposta:2**	
Endereço de *hardware* da origem			
Endereço de protocolo da origem			
Endereço de *hardware* do destino (Vazio no pedido)			
Endereço de protocolo			

Hardware: Protocolo LAN ou WAN
Protocolo: Protocolo da camada de rede

Figura 5.48 Pacote ARP.

o valor desse campo é 1. O campo *tipo de protocolo* define o protocolo da camada de rede: o valor para o protocolo IPv4 é $(0800)_{16}$. Os campos de *endereço de hardware da origem* e *endereço de protocolo da origem* têm tamanho variável e definem os endereços da camada de enlace e da camada de rede do remetente. Os campos de *endereço de hardware do destino* e *endereço de protocolo do destino* definem os endereços da camada de enlace e da camada de rede do destinatário. Um pacote ARP é encapsulado diretamente em um quadro da camada de enlace de dados. O quadro precisa ter um campo para mostrar que a carga útil corresponde ao ARP e não a um datagrama da camada de rede.

Exemplo 5.17

Uma estação cujo endereço IP é I_1 e cujo endereço MAC é E_1 deseja enviar um pacote para outra estação, cujo endereço IP é I_2 e cujo endereço físico é E_2 (o qual não é conhecido pela primeira estação). As duas estações estão na mesma rede. Descreva o pedido ARP e os pacotes de resposta encapsulados em quadros Ethernet (ver Figura 5.55).

Solução

A Figura 5.49 mostra o pedido ARP e os pacotes de resposta. Observe que o campo de dados do ARP nesse caso é de 28 *bytes*, e que os endereços individuais não se encaixam nos limites de 4 *bytes*. Por isso não mostramos os limites usuais de 4 *bytes* para esses endereços. Perceba também que os endereços IP são mostrados em hexadecimal.

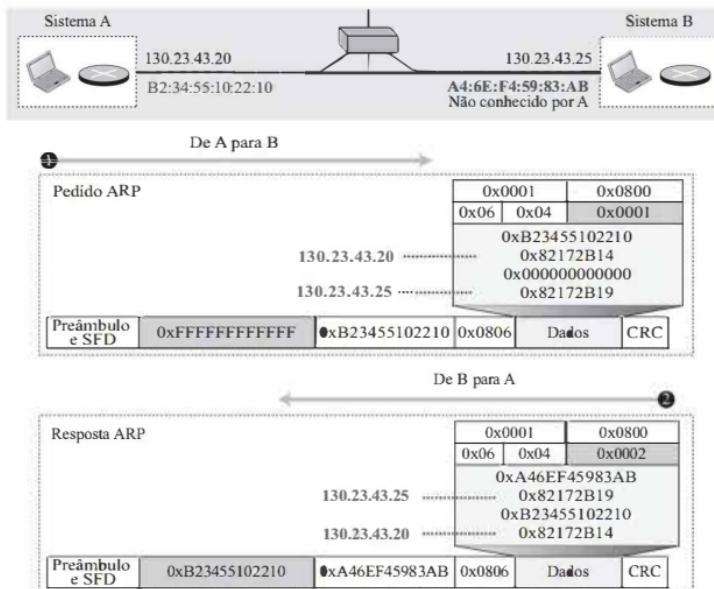



Figura 5.49 Esquema para o Exemplo 5.17.

Um exemplo

Discutimos endereçamento nas quatro camadas mais altas da pilha de protocolos TCP/IP. Não existe endereçamento na camada física, pois a unidade de comunicação nela é um *bit*, o qual não pode carregar um endereço; na camada física, os *bits* são enviados e recebidos por qualquer receptor conectado ao remetente (*broadcast*). Consideramos que agora é a hora de mostrar todos esses endereços em uma única transação, enfatizando o relacionamento entre eles. Também mostramos como todos eles podem de fato ser criados pelo sistema. O que é necessário para a transação é apenas o nome do local que um usuário deseja contatar. A Figura 5.50 mostra uma internet à qual Alice e Bob estão conectados. Alice é uma usuária que quer acessar a lista dos produtos oferecidos por Bob, que tem uma pequena empresa chamada ***Bob.biz***, na qual a lista dos produtos contendo suas especificações é armazenada em um documento denominado ***produtos***.

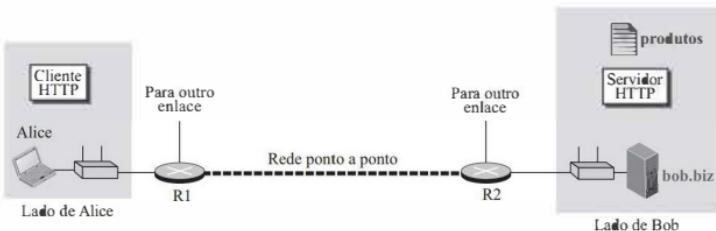



Figura 5.50 A internet para o nosso exemplo.

O único identificador que Alice precisa saber é o URL do *site* de Bob: ***http://bob.biz/product***. O restante dos endereços, de origem e de destino, é gerado pelo sistema, conforme mostra o exemplo.

Operações no lado de Alice

Consideramos que Alice está conectada a uma LAN e que seu computador conhece seu próprio endereço IP e endereço da camada de enlace. O servidor de Bob também está conectado a uma LAN e seu computador conhece seu próprio endereço IP e endereço da camada de enlace. Os roteadores na internet também conhecem seus próprios endereços IP e da camada de enlace. Utilizaremos endereços simbólicos para tornar as figuras mais legíveis. No entanto, Alice não precisa saber qualquer um desses endereços: eles são criados automaticamente à medida que a transação é executada.

A Figura 5.51 mostra o que acontece no lado de Alice. Alice, usando um navegador, digita o URL do *site* do Bob (http://bob.biz/product). O computador de Alice usa o cliente HTTP para responder à solicitação de Alice. O cliente HTTP, no entanto, não sabe o endereço IP de Bob. Esse cliente invoca o cliente DNS para obter ajuda. O cliente DNS envia uma solicitação a um servidor DNS e descobre o endereço IP do computador de Bob ($\mathbf{I_B}$), que é passado para a camada de transporte.

A camada de transporte cria, então, um segmento utilizando o número de porta bem conhecido do servidor HTTP ($\mathbf{P_B}$ = 80) e um número de porta temporário recebido do sistema operacional para o cliente HTTP ($\mathbf{P_A}$). A camada de transporte pode agora criar um segmento e passá-la para a camada de rede com o endereço IP do computador de Bob ($\mathbf{I_B}$), obtido no passo anterior. Obviamente, não mostramos todos os processos que ocorrem no estabelecimento da conexão; consideramos que a conexão é feita aqui e que apenas um segmento é trocado.

Fluxo de pacotes no computador de Alice

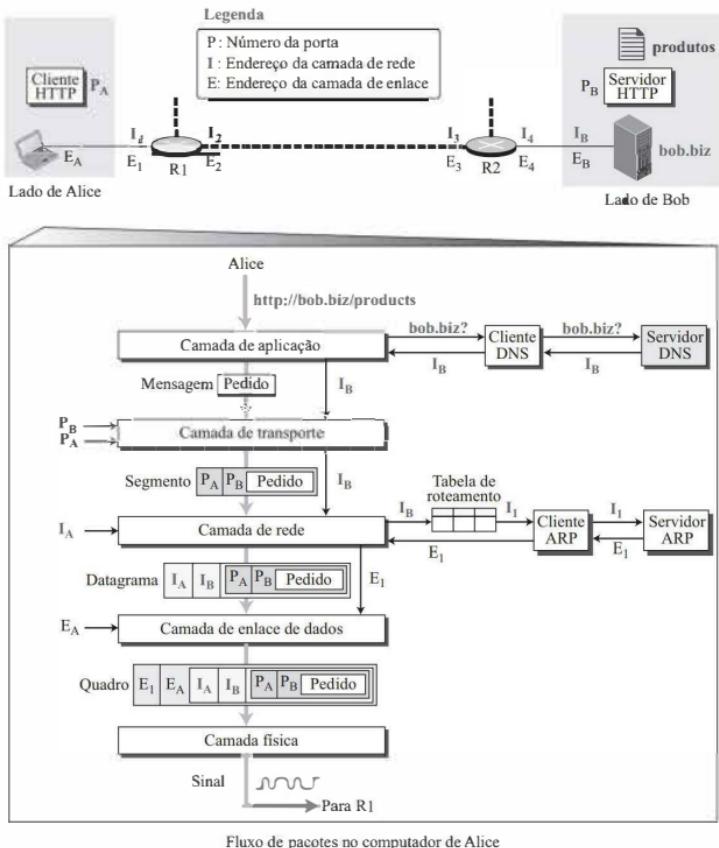



Figura 5.51 Fluxo de pacotes no computador de Alice.

A camada de rede recebe o segmento e $\mathbf{I}_B$, mas precisa descobrir o endereço da camada de enlace. A camada de rede consulta sua tabela de roteamento e tenta descobrir qual é o próximo roteador (nesse caso, o roteador-padrão) para o destino $\mathbf{I}_B$. A tabela de roteamento fornece $\mathbf{I}_1$, mas a camada de rede precisa determinar o endereço da camada de enlace do roteador R1. Ela usa seu cliente ARP para determinar o endereço da camada de enlace E1. A camada de rede pode agora passar o datagrama com o endereço da camada de enlace para a camada de enlace de dados.

A camada de enlace de dados conhece seu próprio endereço de camada de enlace, $\mathbf{E}_A$. Ela cria o quadro e o passa para a camada física, onde os dados são convertidos para sinais (conforme veremos no Capítulo 7) e enviados pelo meio de comunicação.

Operações no roteador R1

Agora, veremos o que acontece no roteador R1. O roteador R1, como sabemos, tem apenas três camadas inferiores. O pacote recebido precisa subir essas três camadas e depois descer A Figura 5.52 mostra essas operações. Na chegada dos dados, a camada física da conexão da esquerda cria o quadro e o passa para a camada de enlace de dados, que desencapsula o datagrama e o passa para a camada de rede. Esta examina o endereço de camada de rede do datagrama e descobre que ele deve ser entregue para o dispositivo cujo endereço IP é $\mathbf{I}_B$. A camada de rede consulta sua tabela de roteamento para descobrir qual é o próximo nó (roteador) no caminho até $\mathbf{I}_B$. A tabela de roteamento retorna $\mathbf{I}_3$ como resultado. Este é o endereço IP do roteador R2, que está no mesmo enlace que R1. A camada de rede usa, então, o cliente e o servidor ARP para determinar o endereço da camada de enlace desse roteador, que vem a ser $\mathbf{E}_3$. A camada de rede passa o datagrama e $\mathbf{E}_3$ para a camada de enlace de dados pertencente à conexão do lado direito. A camada de enlace encapsula o datagrama, adiciona $\mathbf{E}_3$ e $\mathbf{E}_2$ (o seu próprio endereço da camada de enlace) e passa o quadro para a camada física. A camada física codifica os *bits* em sinais e os envia pelo meio de comunicação até R2.

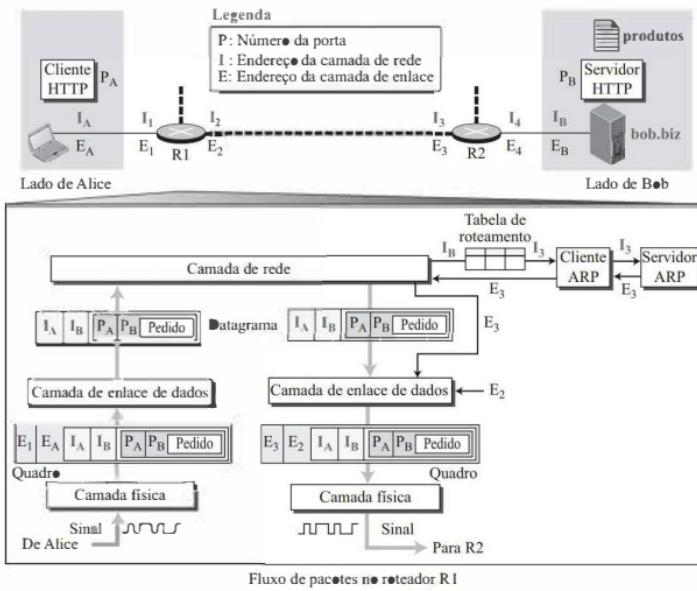



Figura 5.52 Fluxo de operações no roteador R1.

Operações no roteador R2

As operações neste roteador são as mesmas que ocorrem no roteador R1, exceto pelo fato de que alguns endereços são alterados. Por isso, não repetimos tais operações aqui.

Operações no lado de Bob

Agora, veremos o que acontece no lado de Bob. A Figura 5.53 mostra como os sinais no lado de Bob são transformados em uma mensagem para o programa-servidor HTTP. No lado de Bob, não

são mais necessários endereços ou mapeamentos. O sinal recebido do enlace é transformado em um quadro. Este é passado para a camada de enlace de dados, que desencapsula o datagrama e o passa para a camada de rede. Ela desencapsula a mensagem e a passa para a camada de transporte. A camada de transporte desencapsula o segmento e o passa para a porta P_B (80). O servidor HTTP recebe a mensagem, prepara uma cópia do arquivo de produtos, formata-o como um documento HTML, e envia o resultado para Alice usando o mesmo fluxo de comunicação, mas na ordem inversa. A mensagem, provavelmente, passará pelos mesmos roteadores, R1 e R2, até chegar a Alice.

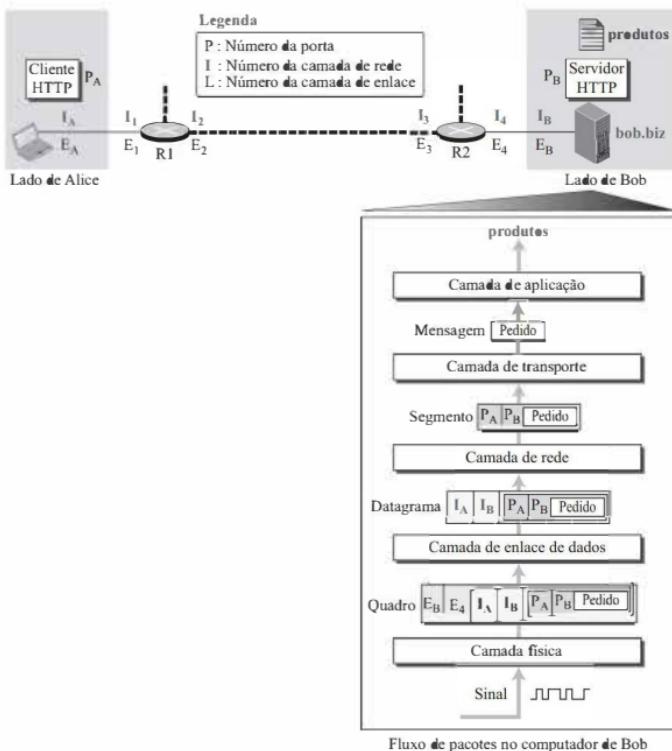



Figura 5.53 Operações no lado de Bob.

5.5 REDES COM FIOS: PROTOCOLO ETHERNET

No Capítulo 1, mencionamos que a pilha de protocolos TCP/IP não define qualquer protocolo da camada de enlace de dados nem da camada física. Em outras palavras, a pilha de protocolos TCP/IP aceita qualquer protocolo para essas duas camadas que sejam capazes de fornecer serviços para a camada de rede. A camada de enlace de dados e a camada física são realmente o território das redes

locais e de longa distância. Isto significa que, quando discutimos essas duas camadas, estamos falando sobre as redes que as estão usando. Como podemos ver neste capítulo e no próximo, podemos ter redes com ou sem fios. Discutimos redes com fios neste capítulo e deixamos a discussão sobre redes sem fios para o próximo capítulo.

No Capítulo 1, aprendemos que uma rede local (LAN) é uma rede de computadores projetada para cobrir uma área geográfica limitada, como um prédio ou um *campus*. Embora uma LAN possa ser usada como uma rede isolada para conectar computadores em uma organização com o único propósito de compartilhar recursos, a maioria das LANs hoje também está ligada a uma rede de longa distância (WAN) ou à Internet.

Nas décadas de 1980 e 1990, vários tipos diferentes de LANs eram utilizados. Todas essas LANs usavam um método de acesso ao meio para resolver o problema do compartilhamento do meio de comunicação. A Ethernet adotou a abordagem CSMA/CD. As tecnologias Token Ring, Token Bus e Interface de Dados Distribuídos por Fibra (FDDI – Fiber Distributed Data Interface) adotaram a abordagem de passagem de ficha. Durante esse período, outra tecnologia de LAN, denominada ATM e que implantou a tecnologia de WAN de alta velocidade (ATM), apareceu no mercado.

Exceto pela Ethernet, quase todas as tecnologias de LAN desapareceram do mercado porque a Ethernet foi capaz de se atualizar para atender as necessidades de cada época. Várias razões para esse sucesso são mencionadas na literatura, mas acreditamos que o protocolo Ethernet foi projetado para que ele fosse capaz de evoluir com a demanda por maiores taxas de transmissão. É natural que uma organização que usava uma LAN Ethernet no passado e agora precisava de uma taxa de dados mais elevada preferisse fazer a atualização para a nova geração em vez de migrar para outra tecnologia, algo que pode envolver maiores custos. Isto significa que limitaremos nossa discussão sobre LANs com fios a uma discussão sobre a Ethernet.

A tecnologia de LAN Ethernet foi desenvolvida em 1970 por Robert Metcalfe e David Boggs. Desde então, ela passou por quatro gerações: **Ethernet Padrão** (10 Mbps), **Fast Ethernet** (100 Mbps), **Ethernet Gigabit** (1 Gbps) e **10 Gibabit Ethernet** (10 Gbps).

5.5.1 Projeto IEEE 802

Antes de discutirmos o protocolo Ethernet e todas as suas gerações, precisamos discutir o padrão IEEE que comumente encontramos na literatura e na vida real. Em 1985, a Sociedade de Computação do IEEE iniciou um projeto, denominado **Projeto 802**, com o objetivo de estabelecer normas para permitir a intercomunicação entre equipamentos de vários fabricantes. O Projeto 802 não tem intenção de substituir qualquer parte do modelo OSI ou da pilha de protocolos TCP/IP. Na verdade, ele é uma maneira de especificar as funções da camada física e da camada de enlace de dados dos principais protocolos de LAN.

A relação do Padrão 802 com a pilha de protocolos TCP/IP é mostrada na Figura 5.54. O IEEE subdividiu a camada de enlace de dados em duas subcamadas: **Controle de Enlace**

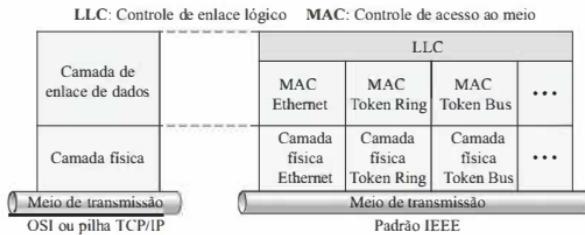



Figura 5.54 Padrões IEEE para LANs.

Lógico (LLC – Logical Link Control) e **Controle de Acesso ao Meio** (MAC – Media Access Control). O IEEE também criou vários padrões de camada física para diferentes protocolos de LAN.

Controle de Enlace Lógico

Anteriormente, discutimos sobre o *controle de enlace de dados*. Dissemos que o controle de enlace de dados lida com o enquadramento, controle de fluxo e controle de erros. No Projeto IEEE 802, as tarefas de controle de fluxo, controle de erros e parte das funções de enquadramento são agrupadas em uma subcamada denominada *Controle de Enlace Lógico* (LLC – Logical Link Control). O enquadramento é tratado tanto na subcamada LLC como na subcamada MAC.

O LLC fornece um único protocolo de controle da camada de enlace para todas as LANs IEEE. Isto significa que o protocolo LLC pode fornecer interconexão entre LANs diferentes porque ele torna a subcamada MAC transparente.

Controle de Acesso ao Meio

Anteriormente, discutimos diversos métodos de acesso, incluindo acesso aleatório, acesso controlado e canalização. O Projeto IEEE 802 criou uma subcamada denominada Controle de Acesso ao Meio (MAC) que define o método de acesso específico para cada LAN. Por exemplo, ela define o CSMA/CD como método de acesso ao meio para LANs Ethernet e define o método de passagem de ficha para LANs Token Ring e Token Bus. Conforme mencionamos na seção anterior, parte da função de enquadramento também é tratada pela camada MAC.

5.5.2 Ethernet Padrão

Referimo-nos à tecnologia Ethernet original com uma taxa de transferência de dados de 10 Mbps como Ethernet Padrão. Embora a maioria das implementações já tenha migrado para outras tecnologias ao longo da evolução da Ethernet, existem algumas características da Ethernet Padrão que não foram alteradas durante essa evolução. Discutimos essa versão básica com o objetivo de preparar o terreno para a compreensão das outras três tecnologias.

Formato dos quadros

O quadro Ethernet contém sete campos, conforme mostra a Figura 5.55.

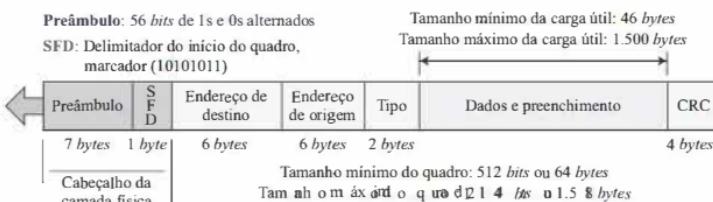



Figura 5.55 Quadro Ethernet.

- **Preâmbulo.** Campo que contém 7 *bytes* (56 *bits*) de 0s e 1s alternados, os quais alertam o sistema receptor da chegada de um quadro e permitem que ele sincronize seu relógio caso ele esteja fora de sincronismo. A sequência de *bits* fornece apenas um alerta e um pulso de

sincronismo. A sequência de 56 *bits* permite que as estações percam alguns *bits* no início do quadro. O *preâmbulo* é, na verdade, adicionado na camada física e não é (formalmente) parte do quadro.

- **Delimitador de Início do Quadro.** Campo de 1 *byte*, com o valor $(10101011)_2$, que indica o início do quadro. O campo SFD (do inglês Start Frame Delimiter) alerta a estação ou estações de que esta é a última chance para realizar a sincronização. Os últimos 2 *bits* são $(11)_2$ e alertam o receptor de que o campo seguinte é o endereço de destino. Esse campo é, na verdade, um marcador que sinaliza o início do quadro. Precisamos ter em mente que um quadro Ethernet é um quadro de tamanho variável. É preciso adicionar um marcador para sinalizar o início do quadro. O campo SFD também é adicionado na camada física.
- **Endereço de destino.** Campo que tem seis *bytes* (48 *bits*), o endereço de destino (ED) contém o endereço da camada de enlace da estação (ou estações) de destino que receberá o pacote. Discutiremos o endereçamento em breve. Quando o receptor vê o seu próprio endereço da camada de enlace, um endereço *multicast* para um grupo do qual o receptor é membro, ou um endereço de *broadcast*, ele desencapsula os dados do quadro e os passa para o protocolo da camada superior definido pelo valor do campo de *tipo*.
- **Endereço de origem.** Campo que também tem seis *bytes*, o endereço de origem (EO) contém o endereço da camada de enlace do remetente do pacote. Vamos discutir o endereçamento em breve.
- **Tipo.** Campo que define o protocolo da camada superior cujo pacote está encapsulado no quadro. Esse protocolo pode ser IP, ARP, OSPF e assim por diante. Em outras palavras, ele serve ao mesmo propósito que o campo de protocolo em um datagrama e que o número da porta em um segmento ou datagrama de usuário. É utilizado para permitir a multiplexação e demultiplexação.
- **Dados.** Campo que transporta os dados encapsulados provenientes dos protocolos da camada superior. Seu comprimento mínimo é de 46 *bytes* e o máximo é de 1.500 *bytes*. Discutiremos a razão para esses valores mínimos e máximos em breve. Se os dados provenientes da camada superior têm um comprimento maior que 1.500 *bytes*, eles devem ser fragmentados e encapsulados em mais de um quadro. Se o tamanho for inferior a 46 *bytes*, ele precisa ser complementado com 0s extras (*padding*). Um quadro de dados preenchido é entregue ao protocolo da camada superior sem alterações (sem remover os *bits* de enchimento), o que significa que a camada superior é responsável por remover ou adicionar o preenchimento. O protocolo da camada superior precisa saber o comprimento dos seus dados. Por exemplo, um datagrama tem um campo que define o tamanho dos dados.
- **CRC.** O último campo contém informações para a detecção de erros, neste caso, um CRC-32. O CRC é calculado sobre os campos de endereços, tipo e dados. Se o receptor calcular o CRC e identificar que ele não vale zero (corrompido na transmissão), ele descarta o quadro.

Serviço não orientado à conexão e não confiável

A Ethernet fornece um serviço não orientado à conexão, o que significa que cada quadro enviado é independente do quadro anterior e do seguinte. A Ethernet não apresenta qualquer fase de estabelecimento de conexão ou de encerramento de conexão. O remetente envia um quadro sempre que ele o tiver, de modo que o receptor pode ou não estar pronto para recebê-lo. O remetente pode sobrecarregar o receptor com quadros, o que pode resultar no descarte de quadros. Se um quadro for descartado, o remetente não será informado sobre o ocorrido. Como o IP, que usa os serviços da Ethernet, também não é orientado à conexão, ele também não será informado sobre o ocorrido. Se a camada de transporte também for um protocolo não orientado à conexão, como é o caso do UDP,

o quadro é perdido e sua recuperação só poderá ser possível se realizada pela camada de aplicação. Entretanto, se a camada de transporte estiver usando TCP, o TCP no remetente não receberá a confirmação para o seu segmento e o enviará novamente.

A Ethernet também não é confiável, assim como ocorre com o IP e com o UDP. Se um quadro for corrompido durante a transmissão e o receptor descobrir a corrupção, o que acontece com alta probabilidade devido ao CRC-32, o receptor descarta o quadro silenciosamente. Perceber o ocorrido faz parte dos deveres dos protocolos das camadas mais altas.

Tamanho dos quadros

A Ethernet impôs restrições sobre os tamanhos mínimo e máximo de um quadro. A restrição do tamanho mínimo é necessária para o correto funcionamento do CSMA/CD, como veremos mais adiante. Um quadro Ethernet deve ter um comprimento mínimo de 512 *bits* ou 64 *bytes*. Parte desse comprimento refere-se ao cabeçalho e ao rodapé. Se contarmos os 18 *bytes* de cabeçalho e rodapé (6 *bytes* de endereço de origem, 6 *bytes* de endereço de destino, 2 *bytes* de tipo e 4 *bytes* de CRC), então o comprimento mínimo dos dados provenientes da camada superior é de 64 – 18 = 46 *bytes*. Se o pacote da camada superior tiver um tamanho inferior a 46 *bytes*, ele é preenchido com *bytes* adicionais para compensar a diferença.

A norma define que o comprimento máximo de um quadro (sem o preâmbulo ou o campo SFD) é de 1.518 *bytes*. Se subtrairmos os 18 *bytes* do cabeçalho e do rodapé, o comprimento máximo da carga útil é de 1.500 *bytes*. A restrição do comprimento máximo tem duas razões históricas: em primeiro lugar, memória era um recurso muito caro quando a Ethernet foi concebida; uma restrição de comprimento máximo ajudava a reduzir o tamanho dos *buffers* (unidades de armazenamento temporário). Em segundo lugar, a restrição de comprimento máximo impede que uma estação monopolize o meio compartilhado, bloqueando outras estações que tenham dados para enviar

Tamanho mínimo do quadro: 64 *bytes*	**Tamanho mínimo dos dados: 46 *bytes***
Tamanho máximo do quadro: 1.518 *bytes*	**Tamanho máximo dos dados: 1.500 *bytes***

Endereçamento

Cada estação em uma rede Ethernet (por exemplo, um PC, uma estação de trabalho ou uma impressora) tem a sua própria **placa de rede**, também conhecida como **Placa de Interface de Rede** (NIC – Network Interface Card). A NIC está inserida na estação e fornece a ela um endereço da camada de enlace. O endereço Ethernet apresenta 6 *bytes* (48 *bits*), normalmente escritos em *notação hexadecimal*, com dois pontos entre cada *byte*. Por exemplo, a seguir mostramos um endereço MAC do protocolo Ethernet:

4A:30:10:21:10:1A

Transmissão de *bits* de endereço

A forma como os endereços são colocados na linha é diferente da forma como eles são escritos em notação hexadecimal. A transmissão é da esquerda para a direita, *byte* a *byte*; no entanto, para cada *byte*, o *bit* menos significativo é enviado primeiro e o mais significativo é enviado por último. Isto significa que o *bit* que define se um endereço é *unicast* ou *multicast* chega primeiro ao receptor. Isso ajuda o receptor a determinar imediatamente se o pacote é *unicast* ou *multicast*.

Exemplo 5.18

Mostre como o endereço 47:20:1B:2E:08:EE é enviado.

Solução

O endereço é enviado da esquerda para a direita, *byte* a *byte*; porém, cada *byte* é enviado da direita para a esquerda, *bit* a *bit*, conforme mostrado a seguir:

Hexadecimal	47	20	1B	2E	08	EE
Binário	01000111	00100000	00011011	00101110	00001000	11101110
Transmitido ←	11100010	00000100	11011000	01110100	00010000	01110111

Endereços *unicast*, *multicast* e *broadcast*

Um endereço de origem é sempre um endereço *unicast* – o quadro vem de uma única estação. O endereço de destino, entretanto, pode ser *unicast*, *multicast* ou *broadcast*. A Figura 5.56 mostra como distinguir um endereço *unicast* de um endereço *multicast*. Se o *bit* menos significativo do primeiro *byte* de um endereço de destino for 0, o endereço é *unicast*; caso contrário, ele é *multicast*.

Figura 5.56 Endereços *unicast* e *multicast*.

Perceba que, da forma como os *bits* são transmitidos, o *bit* que diferencia o *unicast* do *multicast* é o primeiro *bit* transmitido e recebido. O endereço *broadcast* é um caso especial do endereço *multicast*: os destinatários são todas as estações na LAN. Um endereço de destino *broadcast* é formado por 48 *bits* 1.

Exemplo 5.19

Determine o tipo dos seguintes endereços de destino:

a. **4A:30:10:21:10:1A**
b. **47:20:1B:2E:08:EE**
c. **FF:FF:FF:FF:FF:FF**

Solução

Para determinar o tipo de endereço, é preciso inspecionar o segundo dígito hexadecimal a partir da esquerda. Se ele for par, o endereço é de *unicast*. Se for ímpar, o endereço é de *multicast*. Se todos os dígitos forem Fs, o endereço é de *broadcast*. Portanto, temos o seguinte:

a. É um endereço *unicast* porque A em binário é 1010 (par).
b. É um endereço *multicast* porque 7 em binário é 0111 (ímpar).
c. É um endereço *broadcast*, pois todos os dígitos são Fs em hexadecimal.

Distinção entre as transmissões *unicast*, *multicast* e *broadcast*

A Ethernet Padrão usa um cabo coaxial (topologia em barramento) ou um conjunto de cabos de par trançado com um *hub* (topologia em estrela), conforme mostra a Figura 5.57.

(a) Uma LAN com uma topologia em barramento usando um cabo coaxial

(b) Uma LAN em topologia estrela usando um *hub*

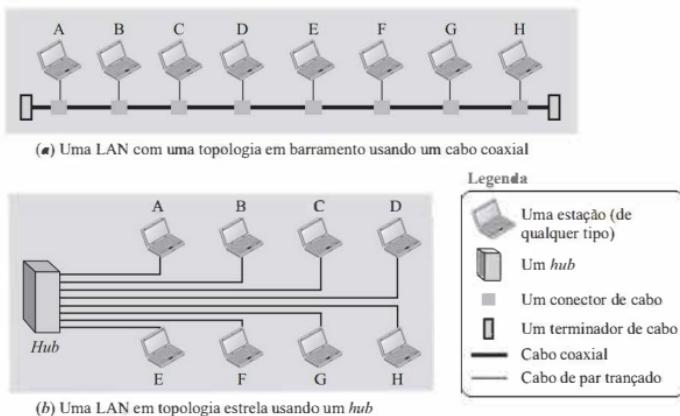



Figura 5.57 Implementação da Ethernet Padrão.

É preciso dizer que a transmissão na Ethernet Padrão é sempre via *broadcast*, não importando se a intenção é fazer *unicast*, *multicast* ou *broadcast*. Em uma topologia em barramento, quando a estação A envia um quadro para a estação B, todas as estações recebem esse quadro. Na topologia em estrela, quando uma estação envia um quadro para a estação B, o *hub* o receberá. Como o *hub* é um elemento passivo, ele não verifica o endereço de destino do quadro; ele regenera os *bits* (caso o sinal tenha se enfraquecido) e os envia para todas as estações com exceção da estação A. Na realidade, ele inunda a rede com o quadro.

A questão, então, é como as transmissões *unicast*, *multicast* e *broadcast* são de fato diferenciadas umas das outras. A resposta está na forma como os quadros são guardados ou descartados.

- Em uma transmissão *unicast*, todas as estações recebem o quadro, mas enquanto o destinatário guarda e manipula esse quadro, as outras estações o descartam.
- Em uma transmissão *multicast*, todas as estações recebem o quadro, mas enquanto as estações que são membros do grupo guardam e manipulam esse quadro, as outras estações o descartam.
- Em uma transmissão *broadcast*, todas as estações (exceto pelo emissor) recebem o quadro e todas elas guardam e manipulam esse quadro.

Método de acesso

Como a rede que usa o protocolo Ethernet Padrão é uma rede de *broadcast*, precisamos usar um método de acesso para controlar o acesso ao meio compartilhado. A Ethernet Padrão adotou o CSMA/CD com o método 1-persistente discutido anteriormente nas Figuras 5.37 a 5.40. Analisaremos um cenário para entender como esse método funciona para o protocolo Ethernet.

- Considere que a estação A na Figura 5.57 tenha um quadro para enviar à estação D. A estação A deve primeiramente verificar se outra estação está transmitindo (detecção de portadora). Esta estação mede o nível de energia no meio (por um curto intervalo de tempo, normalmente menos de 100 μs). Se não houver qualquer energia de sinal no meio,

significa que nenhuma estação está transmitindo (ou que o sinal ainda não atingiu a estação A). A estação A interpreta essa situação como "meio ocioso". Ela, então, começa a enviar seu quadro. Por outro lado, se o nível de energia de sinal não for zero, significa que o meio está sendo usado por outra estação. A estação A monitora continuamente o meio até que ele fique ocioso por 100 μs. Logo em seguida, ela inicia o envio do quadro. No entanto, a estação A precisa manter uma cópia do quadro em seu *buffer* até ter certeza de que não houve uma colisão. O que acontece após a estação A ter certeza disso é o assunto que discutiremos a seguir.

- A escuta do meio não se encerra após a estação A começar a enviar o quadro. A estação A precisa enviar dados e ouvir o meio continuamente. Dois casos podem ocorrer:

a. A estação A envia 512 *bits* e nenhuma colisão é detectada nesse período (o nível de energia não ficou acima do nível de energia normal). A estação tem, então, certeza de que o quadro foi enviado e interrompe a escuta do meio. De onde vem o número de 512 *bits*? Se considerarmos que a taxa de transmissão da Ethernet é 10 Mbps, significa que a estação leva 512/(10 Mbps) = 51,2 μs para enviar 512 *bits*. Com a velocidade de propagação em um cabo (2×10^8 metros), o primeiro *bit* poderia ter viajado 10.240 metros (ida) ou apenas 5.120 metros (ida e volta), ter colidido com um *bit* da última estação do cabo, e voltado. Em outras palavras, se uma colisão vier a ocorrer, ela deve acontecer até o momento em que o remetente enviou 512 *bits* (pior caso) e que o primeiro *bit* fez uma viagem de 5.120 metros. Devemos notar que, se a colisão acontecer no meio do cabo, não no final, a estação A percebe a colisão mais cedo e aborta a transmissão. Precisamos também mencionar outra questão. O pressuposto é que o comprimento do cabo é de 5.120 metros, mas na verdade, por projeto, a Ethernet Padrão, impõe uma restrição de 2.500 metros porque é preciso considerar os atrasos encontrados ao longo do caminho. Isso significa que o pior caso é considerado. A ideia toda é que, se a estação A não detectar a colisão antes de enviar 512 *bits*, não deve ter havido qualquer colisão, porque durante esse tempo, o primeiro *bit* já deve ter atingido a extremidade da linha e todas as outras estações sabem que uma estação está enviando dados e abstêm-se de transmitir. Em outras palavras, o problema ocorre quando alguma outra estação (por exemplo, a última estação) começa a transmitir antes que o primeiro *bit* enviado pela estação A atinja aquela estação. A outra estação erroneamente pensa que a linha está livre porque o primeiro *bit* ainda não chegou até ela. O leitor deve observar que a restrição de 512 *bits*, na verdade, ajuda a estação que está transmitindo: ela tem certeza de que nenhuma colisão ocorrerá se ela não acontecer durante a transmissão dos primeiros 512 *bits*, por isso ela pode descartar a cópia do quadro de seu *buffer*.

b. A estação A detecta uma colisão antes de terminar de enviar 512 *bits*. Isto significa que um dos *bits* anteriores colidiu com um *bit* enviado por outra estação. Nesse caso, ambas as estações devem abster-se de enviar e manter os quadros em seus *buffers* para reenviá-los quando a linha voltar a ficar disponível. No entanto, para informar outras estações de que houve uma colisão na rede, as estações envia um sinal de interferência (*jamming*) de 48 *bits*. O objetivo do sinal de interferência é criar sinal suficiente (mesmo se a colisão acontecer após alguns poucos *bits*) para alertar as outras estações sobre a colisão. Depois de enviar o sinal de interferência, as estações precisam incrementar o valor de K (número de tentativas). Se, após ser incrementado, K atingir o valor de 15, a experiência mostrou que a rede está muito ocupada; a estação deve interromper seus esforços e tentar novamente. Se K < 15, a estação pode esperar um tempo de retardo (T_B na Figura 5.40) e reiniciar o processo. Como mostra a Figura 5.40, a estação cria um número aleatório entre 0 e $2^K - 1$, o que significa que cada vez que ocorrer uma colisão, a faixa de valores possíveis para o número aleatório aumenta exponencialmente. Após a primeira colisão (K = 1) o número aleatório

encontra-se na faixa (0,1). Após a segunda colisão (K = 2), ele está na faixa (0, 1, 2, 3). Após a terceira (K = 3), a faixa passa a ser (0, 1, 2, 3, 4, 5, 6, 7). Assim, depois de cada colisão, aumenta a probabilidade de o tempo de retardo ficar maior. Isto acontece pelo fato de que, se a colisão acontecer mesmo após a terceira ou quarta tentativa, significa que a rede está realmente ocupada; um maior tempo de retardo é necessário.

Eficiência da Ethernet Padrão

A eficiência da Ethernet é definida como a razão entre o tempo utilizado por uma estação para enviar dados e o tempo em que o meio é ocupado por ela. A eficiência prática da Ethernet Padrão foi medida como

$$\textbf{Eficiência} = 1 / (1 + 6{,}4 \times a)$$

onde o parâmetro ***a*** é o número de quadros que podem ser colocados simultaneamente no meio. Ele pode ser calculado como ***a*** = (atraso de propagação)/(atraso de transmissão), pois o atraso de transmissão é o tempo necessário para um quadro de tamanho médio ser colocado no meio e o atraso de propagação é o tempo que ele leva para atingir a extremidade do meio. Perceba que, à medida que o valor do parâmetro **a** diminui, a eficiência aumenta. Isto significa que, à medida que o comprimento do meio de comunicação for menor ou o tamanho do quadro for maior, tem-se um aumento da eficiência. No caso ideal, ***a*** = 0 e a eficiência é 1. Pedimos para o leitor calcular essa eficiência nos problemas do fim do capítulo.

Exemplo 5.20

Na Ethernet Padrão, com a taxa de transmissão de 10 Mbps, consideramos que o comprimento do meio é de 2.500 m e que o tamanho do quadro é de 512 *bits*. A velocidade de propagação de um sinal em um cabo é normalmente de 2×10^8 m/s.

Atraso de propagação = $2500/(2 \times 10^8) = 12{,}5\ \mu s$	Atraso de transmissão = $512/(10^7) = 51{,}2\ \mu s$
a = 12,5/51,2 = 0,24	Eficiência = 39%

O exemplo mostra que *a* = 0,24, o que significa que apenas 0,24 de um quadro ocupa o meio todo nesse caso. A eficiência é de 39%, considerado um valor moderado; isto significa que em apenas 61% do tempo o meio está ocupado, mas não está sendo utilizado por uma estação.

Implementação

O protocolo Ethernet Padrão definia várias implementações possíveis, mas apenas quatro delas se tornaram populares durante a década de 1980. A Tabela 5.6 mostra um resumo das implementações da Ethernet Padrão.

Tabela 5.6 Resumo das implementações da Ethernet Padrão.

Implementação	Meio	Comprimento do meio	Codificação
10Base5	Cabo coaxial grosso	500 m	Manchester
10Base2	Cabo coaxial fino	185 m	Manchester
10Base-T	2 UTPs	100 m	Manchester
10Base-F	2 Fibras	2000 m	Manchester

Na nomenclatura 10BaseX, o prefixo numérico define a taxa de transferência de dados (10 Mbps), o termo *Base* significa sinal (digital) de banda base e X define aproximadamente o comprimento máximo do cabo em unidades de 100 metros (por exemplo, 5 para 500 ou 2 para 185 metros) ou o tipo do cabo, T para cabo de par trançado sem blindagem (UTP – Unshielded Twisted Pair) e F para fibra óptica. A Ethernet Padrão utiliza um sinal de banda base, o que significa que os *bits* são transformados em um sinal digital e enviados diretamente para a linha. Todas as implementações usam codificação Manchester, que será discutida em detalhes no Capítulo 7.

5.5.3 Fast Ethernet (100 Mbps)

Na década de 1990, algumas tecnologias de LAN com taxas de transmissão superiores a 10 Mbps, como FDDI e Fiber Channel, apareceram no mercado. Se o padrão Ethernet quisesse sobreviver, ele seria obrigado a competir com essas tecnologias. A Ethernet deu um grande salto, aumentando a taxa de transmissão para 100 Mbps, e a nova geração foi chamada de Fast Ethernet ou Ethernet Rápida. Os projetistas da Fast Ethernet precisavam torná-la compatível com a Ethernet Padrão. A subcamada MAC manteve-se inalterada, o que significava que o formato do quadro e os tamanhos máximo e mínimo também podiam permanecer inalterados. Devido ao aumento da taxa de transmissão, características da Ethernet Padrão que dependiam da taxa de transmissão, método de acesso e implementação, tiveram que ser reconsideradas.

Método de acesso

Lembre-se que o funcionamento correto do CSMA/CD depende da taxa de transmissão, do tamanho mínimo do quadro e do comprimento máximo da rede. Se quisermos conservar o tamanho mínimo do quadro, o comprimento máximo da rede deve ser alterado. Em outras palavras, se o tamanho mínimo do quadro continua sendo de 512 *bits*, e ele for transmitido 10 vezes mais rápido, a colisão precisa ser detectada 10 vezes mais cedo, ou seja, o comprimento máximo da rede deve ser 10 vezes menor (a velocidade de propagação não se altera). Assim, a Fast Ethernet surgiu com duas soluções (ela funciona com qualquer uma destas opções):

1. A primeira solução era remover completamente a topologia de barramento; nesse caso, utiliza-se um *hub* passivo e uma topologia em estrela, mas reduz-se o tamanho máximo da rede para 250 metros em vez dos 2.500 metros da Ethernet Padrão. Essa abordagem mantém a compatibilidade com a Ethernet Padrão.
2. A segunda solução é usar um *switch* de camada de enlace (discutido mais adiante neste capítulo) com *buffers* para armazenar quadros e com conexão *full-duplex* para cada estação, de modo a criar um meio de transmissão privado para cada estação. Nesse caso, não é necessário utilizar CSMA/CD, pois as estações não competem umas com as outras. O *switch* da camada de enlace recebe um quadro de uma estação de origem e o armazena em seu *buffer* (fila), de modo que ele fica à espera de processamento. Em seguida, o *switch* verifica o endereço de destino do quadro e o envia pela interface correspondente. Como a conexão com o *switch* é *full-duplex*, a estação de destino pode até mesmo enviar um quadro para outra estação ao mesmo tempo em que ela recebe um quadro. Em outras palavras, o meio compartilhado é transformado em diversos meios de comunicação ponto a ponto, não havendo necessidade de mecanismos de contenção.

Autonegociação

Um novo recurso adicionado à Fast Ethernet é a *autonegociação*. Ela fornece a uma estação ou *hub* uma variedade de recursos. A autonegociação permite que dois dispositivos negociem o modo e a taxa de transferência de dados da operação. Esse mecanismo foi concebido especialmente para

permitir que dispositivos incompatíveis possam se conectar uns aos outros. Por exemplo, um dispositivo com uma taxa de transferência de dados máxima de 10 Mbps pode se comunicar com um dispositivo com uma taxa de transferência de dados de 100 Mbps (desde que este último seja capaz de operar a uma taxa mais baixa).

Implementação

A implementação da Fast Ethernet na camada física pode ser categorizada dependendo se ela usa dois ou quatro fios. A implementação de dois fios pode consistir em um Par Trançado Blindado (STP – Shielded Twisted Pair), denominado 100BASE-TX, ou em um cabo de fibra óptica, denominado 100Base-FX. A implementação de quatro fios foi concebida apenas para uso com Par Trançado Sem Blindagem (UTP – Unshielded Twisted Pair), conhecido como 100Base-T4. A Tabela 5.7 é um resumo das implementações da Fast Ethernet. Discutiremos as codificações no Capítulo 7.

Tabela 5.7 Resumo de implementações da Fast Ethernet.

Implementação	Meio	Comprimento do meio	Número de fios	Codificação
100Base-TX	STP	100 m	2	4B5B + MLT-3
100Base-FX	Fibra	185 m	2	4B5B + NRZ-I
100Base-T4	UTP	100 m	4	Dois 8B/6T

5.5.4 Ethernet Gigabit

A necessidade de uma taxa de transferência de dados ainda mais elevada levou ao projeto do protocolo Ethernet Gigabit (1000 Mbps). O comitê do IEEE batizou o padrão de 802.3z. O objetivo da Ethernet Gigabit era aumentar a taxa de transferência de dados para 1 Gbps, mas mantendo o mesmo comprimento de endereço, o formato do quadro e seus comprimentos máximo e mínimo.

Subcamada MAC

Uma das principais considerações na evolução da Ethernet era manter inalterada a subcamada MAC. No entanto, para atingir uma taxa de transferência de dados de 1 Gbps, isto já não era mais possível. A Ethernet Gigabit adota duas abordagens distintas para acesso ao meio: *half-duplex* e *full-duplex*. Quase todas as implementações da Ethernet Gigabit seguem a abordagem *full-duplex* e, por isso, não discutimos o modo *half-duplex*. No modo *full-duplex*, existe um *switch* central conectado a todos os computadores ou outros *switches*. Nesse modo, para cada porta de entrada, cada *switch* tem *buffers* nos quais os dados são armazenados até serem transmitidos. Como o *switch* usa o endereço de destino do quadro e o envia pela porta conectada a este destino em particular, não há colisões. Isto significa que o CSMA/CD não é utilizado. A ausência de colisões implica que o comprimento máximo de um cabo é determinado pela atenuação do sinal neste cabo, não pelo processo de detecção de colisões.

Implementação

A Tabela 5.8 é um resumo das implementações da Ethernet Gigabit. O-C e O-L significam onda curta e onda longa, respectivamente. Discutimos questões de codificação no Capítulo 7.

Tabela 5.8 Resumo das implementações da Ethernet Gigabit.

Implementação	Meio	Comprimento do meio	Número de fios	Codificação
1000Base-SX	Fibra O-C	550 m	2	8B/10B + NRZ
1000Base-LX	Fibra O-L	5000 m	2	8B/10B + NRZ
1000Base-CX	STP	25 m	2	8B/10B + NRZ
1000Base-T4	UTP	100 m	4	4D-PAM5

5.5.5 Ethernet 10-Gigabit

Nos últimos anos, a Ethernet passou a ser revista para o uso em áreas metropolitanas. A ideia é estender a tecnologia, a taxa de transferência de dados e a distância de cobertura, de modo que a Ethernet possa ser usada como LAN e MAN (*Metropolitan Area Network*, ou Rede de Área Metropolitana). O comitê do IEEE criou a Ethernet 10-Gigabit e a batizou de Padrão 802.3ae. Os objetivos de projeto da Ethernet 10-Gigabit podem ser resumidos como a elevação da taxa de transferência de dados para 10 Gbps, a manutenção do mesmo tamanho e formato dos quadros, e a possibilidade de interconexão de LANs, MANs e WANs. Atualmente, essa taxa de transferência de dados só é possível com a tecnologia de fibra óptica. O padrão define dois tipos de camadas físicas: LAN PHY e WAN PHY. A primeira é projetada para suportar LANs existentes; já a segunda, na verdade, define uma WAN com enlaces conectados usando SONET OC-192 (discutido mais adiante).

Implementação

A Ethernet 10-Gigabit funciona apenas em modo *full-duplex*, o que significa que não há necessidade de mecanismos de contenção; o CSMA/CD não é usado na Ethernet 10-Gigabit. Quatro implementações são as mais comuns: 10GBase-SR, 10GBase-LR, 10GBase-EW e 10GBase-X4. A Tabela 5.9 mostra um resumo das implementações da Ethernet 10-Gigabit. Discutiremos questões de codificação no Capítulo 7.

Tabela 5.9 Resumo das implementações da Ethernet 10-Gigabit.

Implementação	Meio	Comprimento do meio	Número de fios	Codificação
10GBase-SR	Fibra de 850 nm	300 m	2	64B66B
10GBase-LR	Fibra de 1310 nm	10 Km	2	64B66B
10GBase-EW	Fibra de 1350 nm	40 Km	2	SONET
10GBase-X4	Fibra de 1310 nm	300 m a 10 Km	2	8B10B

5.5.6 LANs virtuais

Uma estação é considerada parte de uma LAN se ela pertencer fisicamente àquela LAN. O critério de participação na LAN é geográfico. O que acontece se precisarmos de uma conexão virtual entre duas estações pertencentes a duas LANs físicas distintas? Basicamente, podemos definir uma **Rede Local Virtual** (VLAN – Virtual Local Area Network) como uma rede local configurada por *software*, e não pelo cabeamento físico.

Usaremos um exemplo para elaborar melhor essa definição. A Figura 5.58 mostra uma LAN comutada em uma empresa de engenharia na qual dez estações são agrupadas em três LANs conectadas por um *switch*.

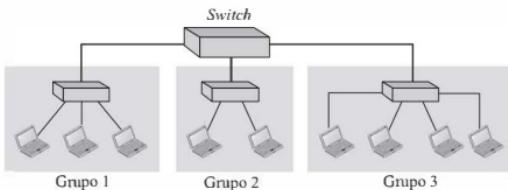



Figura 5.58 Um *switch* conectando três LANs.

Os primeiros três engenheiros trabalham juntos como o primeiro grupo, os outros dois engenheiros trabalham juntos como o segundo grupo e os últimos quatro engenheiros trabalham juntos como o terceiro grupo. A LAN é configurada para permitir esse arranjo.

Mas o que aconteceria se os administradores precisassem mover dois engenheiros do primeiro grupo para o terceiro grupo com o objetivo de acelerar o projeto que está sendo realizado pelo terceiro grupo? A configuração da LAN precisaria ser alterada. O técnico da rede deverá reconfigurar os fios/cabos. O problema repete-se se, uma semana depois, os dois engenheiros voltarem para seu grupo original. Em uma LAN comutada, mudanças nos grupos de trabalho se traduzem em mudanças físicas na configuração da rede.

A Figura 5.59 mostra a mesma LAN comutada dividida em VLANs. A ideia da tecnologia de VLAN é dividir uma LAN em segmentos lógicos em vez de físicos. A LAN pode ser dividida em várias LANs lógicas denominadas VLANs. Cada VLAN constitui um grupo de trabalho na organização. Se uma pessoa se mover de um grupo para outro, não é necessário alterar a configuração física. A participação em grupos nas VLANs é definida via *software*, não via *hardware*. Qualquer estação pode ser logicamente movida para outra VLAN. Todos os membros pertencentes a uma VLAN podem receber mensagens de *broadcast* enviadas àquela VLAN em particular. Isto significa que, se uma estação é movida da VLAN 1 para a VLAN 2, ela recebe mensagens de *broadcast* enviadas para a VLAN 2, porém não recebe mensagens de *broadcast* enviadas para a VLAN 1.

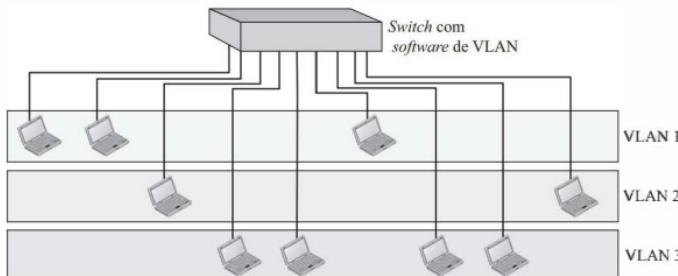



Figura 5.59 Um *switch* usando *software* de VLAN.

É óbvio que o problema ilustrado no exemplo anterior pode ser facilmente resolvido usando VLANs. Mover engenheiros de um grupo para outro via *software* é mais fácil do que alterar a configuração da rede física.

A tecnologia de VLAN permite ainda que estações conectadas a *switches* diferentes sejam agrupadas em uma VLAN. A Figura 5.60 mostra o *backbone* de uma rede local com dois *switches* e três VLANs. Cada VLAN tem estações conectadas tanto ao *switch* A como ao *switch* B.

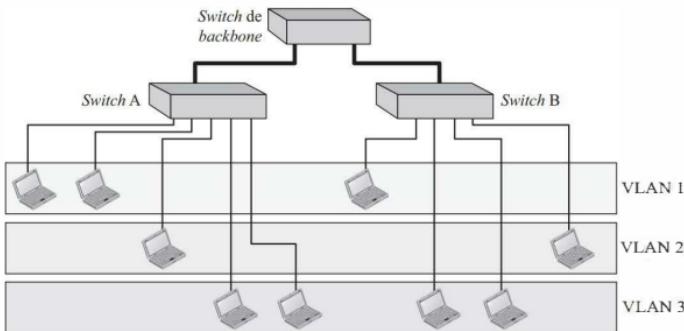



Figura 5.60 Dois *switches* em um *backbone* usando *software* de VLAN.

Essa é uma configuração interessante para uma empresa que possui dois edifícios separados. Cada edifício pode ter sua própria LAN comutada, as quais são conectadas por um *backbone*. As pessoas no primeiro edifício e as pessoas no segundo edifício podem pertencer ao mesmo grupo de trabalho apesar de estarem conectadas a diferentes LANs físicas.

Considerando esses três exemplos, podemos ver que uma VLAN define domínios de *broadcast*. As VLANs agrupam estações pertencentes a uma ou mais LANs físicas em domínios de *broadcast*. As estações em uma VLAN se comunicam umas com as outras como se estivessem conectadas a um mesmo segmento físico.

Associação a grupos

Qual característica pode ser usada para agrupar estações em uma VLAN? Diferentes fornecedores usam diferentes características, tais como número da interface, números de porta, endereços MAC, endereços IP, endereços IP *multicast* ou uma combinação de duas ou mais delas.

Números de interface

Alguns fornecedores de soluções de VLAN usam os números de interface do *switch* como a característica que define a associação das estações aos grupos. Por exemplo, o administrador pode definir que as estações conectadas às portas 1, 2, 3 e 7 pertencem à VLAN 1, que as estações conectadas às portas 4, 10 e 12 pertencem à VLAN 2, e assim por diante.

Endereços MAC

Alguns fornecedores de soluções de VLAN usam o endereço MAC de 48 *bits* como uma característica que define a associação das estações aos grupos. Por exemplo, o administrador pode estipular que as estações com os endereços MAC E2:13:42:A1:23:34 e F2:A1:23:BC:D3:41 pertencem à VLAN 1.

Endereços IP

Alguns fornecedores de soluções de VLAN usam o endereço IP de 32 *bits* (ver Capítulo 4) como uma característica que define a associação das estações aos grupos. Por exemplo, o administrador pode estipular que as estações com os endereços IP 181.34.23.67, 181.34.23.72, 181.34.23.98 e 181.34.23.112 pertencem à VLAN 1.

Endereços IP *multicast*

Alguns fornecedores de soluções de VLAN usam o endereço IP *multicast* (ver Capítulo 4) como uma característica que define a associação das estações aos grupos. O *multicast* na camada IP se traduz então em *multicast* na camada de enlace de dados.

Combinação

Mais recentemente, o *software* disponibilizado por alguns fornecedores passou a permitir combinações de todas essas características. O administrador pode escolher uma ou mais características ao instalar o *software*. Além disso, o *software* pode ser reconfigurado para modificar essas escolhas.

Configuração

Como as estações são agrupadas em VLANs diferentes? As estações são configuradas usando uma de três formas: manual, semiautomática ou automática.

Configuração manual

Na configuração manual, o administrador da rede usa o *software* de VLAN para atribuir manualmente as estações a diferentes VLANs durante a instalação. Mais tarde, a migração de uma VLAN para outra também é feita manualmente. Perceba que não estamos falando de uma configuração física, mas, sim, de uma configuração lógica. O termo *manualmente* aqui significa que o administrador digita os números de portas, os endereços IP ou outras características utilizando o *software* de VLAN.

Configuração automática

Em uma configuração automática, as estações são automaticamente conectadas ou desconectadas de uma VLAN com base em critérios definidos pelo administrador. Por exemplo, o administrador pode especificar o número do projeto como o critério para que uma estação seja membro de um grupo. Quando um usuário muda de projeto, ele automaticamente migra para uma nova VLAN.

Configuração semiautomática

Uma configuração semiautomática é uma solução intermediária entre uma configuração manual e outra automática. Normalmente, a inicialização é feita manualmente, mas as migrações são feitas automaticamente.

Comunicação entre *switches*

Em um *backbone* com múltiplos *switches*, cada *switch* deve saber não apenas qual estação pertence a qual VLAN, mas também a associação de estações conectadas a outros *switches*. Por exemplo, na Figura 5.60, o *switch* A deve conhecer a associação atual das estações conectadas ao *switch* B, e o mesmo vale para o *switch* B com relação ao *switch* A. Três métodos foram concebidos com esse propósito: manutenção de tabela, marcação de quadros e multiplexação por divisão de tempo.

Manutenção de tabela

Neste método, quando uma estação envia um quadro via *broadcast* para os membros do seu grupo, o *switch* cria uma entrada em uma tabela e registra a associação da estação. Os *switches* enviam suas tabelas uns aos outros periodicamente para atualização.

Marcação de quadros

Neste método, enquanto um quadro trafega entre os *switches*, um cabeçalho extra é adicionado a este quadro para especificar a sua VLAN de destino. A marcação do quadro é usada pelos *switches* receptores para determinar as VLANs que devem receber a mensagem de *broadcast*.

Multiplexação por Divisão de Tempo (TDM)

Neste método, a conexão (conhecida como *trunk*) entre os *switches* é dividida em canais compartilhados no tempo (ver a discussão sobre TDM no Capítulo 7). Por exemplo, se o número total de VLANs em um *backbone* for cinco, cada conexão será dividida em cinco canais. O tráfego destinado à VLAN 1 viaja no canal 1, o tráfego destinado à VLAN 2 viaja no canal 2, e assim por diante. O *switch* que recebe o quadro determina a VLAN de destino verificando o canal pelo qual o quadro chegou.

Padrão IEEE

Em 1996, a subcomissão do IEEE 802.1 aprovou um padrão chamado 802.1Q que especifica o formato para a marcação de quadros. O padrão também define o formato a ser utilizado em *backbones* com múltiplos *switches* e permite a utilização de equipamento de fornecedores diferentes na mesma VLAN. O IEEE 802.1Q abriu caminho para uma maior padronização relativa a outras questões envolvendo VLANs. A maioria dos fornecedores já aderiu ao padrão.

Vantagens

Existem diversas vantagens em se usar VLANs.

Redução de tempo e custo

VLANs podem reduzir o custo envolvido na migração de estações de um grupo para outro. A reconfiguração física leva tempo e é custosa. Em vez de mover fisicamente uma estação para outro segmento da rede ou mesmo para outro *switch*, é muito mais fácil e rápido movê-la via *software*.

Criação de grupos de trabalho virtuais

VLANs podem ser usadas para criar grupos de trabalho virtuais. Por exemplo, em um *campus*, professores que trabalham no mesmo projeto podem enviar mensagens de *broadcast* uns aos outros sem a necessidade de pertencerem ao mesmo departamento. Isso pode reduzir tráfego como se a capacidade de *multicast* do IP estivesse sendo usada.

Segurança

VLANs podem ser usadas como uma medida adicional de segurança. Pessoas que pertencem ao mesmo grupo podem enviar mensagens de *broadcast* com a garantia de que os usuários de outros grupos não receberão tais mensagens.

5.6 OUTRAS REDES COM FIOS

Conforme discutimos no Capítulo 1, as redes que encontramos na Internet são classificadas como LANs ou WANs. No entanto, algumas vezes, não há consenso na terminologia. Por exemplo,

algumas redes de acesso, como conexões discadas ou conexões a cabo, são chamadas WANs por algumas pessoas e de Redes Metropolitanas (MANs – Metropolitan Area Networks) por outras. Nesta seção, discutimos algumas dessas redes que são, direta ou indiretamente, usadas na Internet, sem, no entanto, classificá-las como MANs ou WANs; aqui, elas serão denominadas simplesmente redes.

5.6.1 Redes ponto a ponto

Algumas redes ponto a ponto, tais como conexões discadas, DSL e a cabo, são usadas para fornecer acesso à Internet a partir das instalações dos usuários. Como essas redes utilizam uma conexão dedicada entre os dois dispositivos, elas não usam protocolos de Controle de Acesso ao Meio (MAC – Media Access Control). O único protocolo necessário é o PPP, conforme discutimos anteriormente.

Conexão discada

Redes ou conexões discadas usam os serviços fornecidos pelas redes de telefonia para transmitir dados. A rede de telefonia teve o seu início no fim do século 19*. A rede toda, também conhecida como *Rede de Telefonia Convencional* (POTS – Plain Old Telephone System), foi concebida originalmente como um sistema analógico para transmissão de voz. Com o advento da era do computador, a rede, na década de 1980, começou a transportar dados juntamente com voz. Nas últimas décadas, a rede de telefonia passou por diversas alterações técnicas. De fato, a maior parte da rede de telefonia é agora digital. A única parte que continua sendo analógica é a linha que conecta o assinante à rede de telefonia. A necessidade de transmitir dados digitais levou à invenção do modem discado.

O termo **modem** é uma palavra composta que se refere às duas entidades funcionais que compõem o dispositivo: um *modulador* de sinal e um *demodulador* de sinal. Um **modulador** cria um sinal analógico a partir de dados digitais. Um **demodulador** recupera os dados digitais a partir do sinal modulado. Tais conexões podem ser usadas apenas se uma das partes estiver usando sinalização digital (por exemplo, via um provedor de Internet). Elas são assimétricas no sentido de que a taxa de recepção de dados (fluxo de dados vindo do provedor de serviços de Internet para o PC) é no máximo de 56 kbps, enquanto a taxa de envio de dados (fluxo de dados indo do PC para o provedor de Internet) pode atingir um máximo de 33,6 kbps, conforme mostra a Figura 5.61.

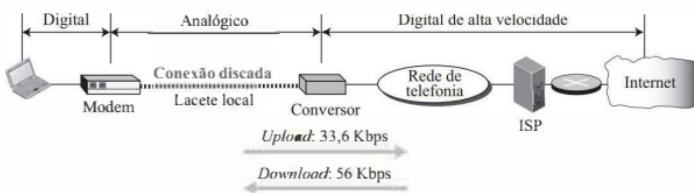



Figura 5.61 Rede discada para prover acesso à Internet.

No *upload* (envio), o sinal analógico ainda precisa ser amostrado antes de entrar na rede de telefonia digital de alta velocidade. Nessa direção, o ruído de quantização (conforme veremos no Capítulo 7) é introduzido no sinal, o que reduz a taxa de transmissão para 33,6 kbps. No entanto, não há qualquer processo de amostragem no *download* (recepção). O sinal não é afetado pelo ruído de quantização. O usuário individual que acessa a Internet normalmente precisa de uma velocidade maior para *download* que para *upload*. Esses usuários desejam, por exemplo, obter um arquivo grande com maior

* N. de T.: O leitor interessado pode encontrar informações sobre a história da rede de telefonia no Brasil no *site* do Ministério das Telecomunicações: http://www.mc.gov.br/o-ministerio/historico/historia-da-telefonia.

frequência do que desejam enviar um arquivo dessa natureza, de modo que a velocidade assimétrica normalmente passa despercebida. É preciso mencionar, entretanto, que quando uma linha de telefone é usada para acesso à Internet, ela não pode ser usada para comunicação de voz ao mesmo tempo.

Podemos nos perguntar como chegamos a uma velocidade de 56 kbps. As empresas de telefonia praticam uma frequência de 8.000 amostras por segundo, com 8 *bits* por amostra. Um dos *bits* em cada amostra é utilizado para fins de controle, o que significa que cada amostra apresenta 7 *bits*. A taxa de transferência é, portanto, 8.000 × 7, ou 56.000 bps ou 56 kbps.

Linha de Assinante Digital

Depois que os modems tradicionais atingiram o seu pico de taxa de transferência de dados, as empresas de telefonia desenvolveram uma outra tecnologia, denominada DSL, para proporcionar maior velocidade de acesso à Internet. A tecnologia **Linha de Assinante Digital** (DSL – Digital Subscriber Line) é uma das mais promissoras no sentido de suportar comunicações digitais de alta velocidade sobre a infraestrutura de telefonia existente. A tecnologia DSL consiste em um conjunto de tecnologias, cada uma com uma primeira letra diferente (ADSL, VDSL, HDSL e SDSL). O conjunto é muitas vezes denominado *x*DSL, onde o *x* pode ser substituído por A, V, H ou S. Aqui, discutimos apenas o primeiro, o ADSL. A primeira tecnologia no conjunto é a *DSL Assimétrica* (*ADSL*). A ADSL, assim como um modem de 56K, fornece maior velocidade (taxa de transferência de *bits*) na direção de recepção (da Internet para a residência) do que na direção de envio (da residência para a Internet). Por isso ela é denominada assimétrica. Contrariamente à assimetria em modems de 56K, os projetistas da ADSL dividiram propositalmente a largura de banda disponível de forma irregular no lacete local para o cliente residencial. O serviço não é adequado para clientes empresariais que necessitam de uma grande largura de banda em ambas as direções.

Usando lacetes locais existentes

Um ponto interessante é que a tecnologia ADSL utiliza as linhas telefônicas existentes (lacete local). Mas como a ADSL alcança uma taxa de dados que nunca foi conseguida com modems tradicionais? A resposta é que o cabo de par trançado usado nas linhas telefônicas é, na verdade, capaz de lidar com uma largura de banda de até 1,1 MHz, mas os filtros instalados na central local da companhia telefônica, à qual todo lacete local se conecta, limitam a largura de banda a 4 kHz (suficiente para comunicações de voz). Se o filtro for removido, no entanto, toda a faixa de 1,1 MHz fica disponível para comunicações de dados e de voz. Tipicamente, uma largura de banda disponível de 1.104 MHz é dividida em um canal de voz, um canal de recepção e um canal de envio, conforme mostra a Figura 5.62.

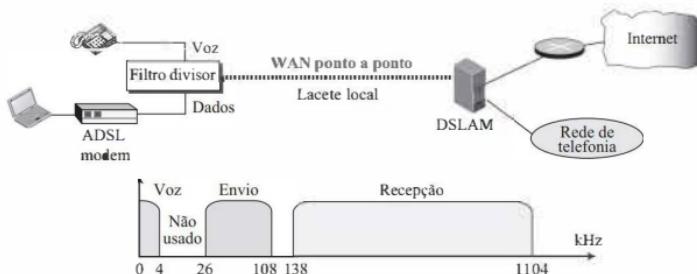



Figura 5.62 Rede (ADSL) ponto a ponto.

A ADSL permite que o assinante use o canal de voz e o canal de dados simultaneamente. A taxa de transferência na direção de envio pode chegar a 1,44 Mbps. Entretanto, a taxa de transferência de dados é normalmente inferior a 500 kbps por causa do elevado nível de ruído nesse canal. A taxa de transferência de dados na direção de recepção pode chegar a 13,4 Mbps. No entanto, esta taxa é normalmente inferior a 8 Mbps por causa do ruído nesse canal. Um ponto muito interessante é que a empresa de telefonia, nesse caso, atua como o ISP, de modo que serviços como e-mail ou acesso à Internet podem ser fornecidos pela companhia de telefonia em si.

Cabo

As redes a cabo foram originalmente criadas para fornecer acesso a programas de TV para assinantes que não conseguiam receber o sinal devido a obstáculos naturais, como montanhas. Mais tarde, as redes a cabo tornaram-se populares entre pessoas que queriam simplesmente um sinal de melhor qualidade. Além disso, as redes a cabo permitiam o acesso a estações remotas de TV por meio de conexões de micro-ondas. As empresas de TV a cabo também descobriram um mercado promissor no fornecimento de acesso à Internet, usando alguns dos canais originalmente projetados para vídeo. Depois de descrevermos a estrutura básica das redes a cabo, discutiremos como os modems a cabo podem fornecer uma conexão de alta velocidade à Internet.

Redes a cabo tradicionais

A TV a cabo começou a distribuir sinais de radiodifusão de vídeo em locais com recepção ruim ou inexistente no final da década de 1940. A tecnologia era chamada **Antena Comunitária de TV** (CATV – Community Antenna TV), porque uma antena no topo de uma colina alta ou edifício recebia os sinais das emissoras de TV e os redistribuía, via cabos coaxiais, para a comunidade. A Figura 5.63 mostra um diagrama esquemático de uma rede de TV a cabo tradicional.

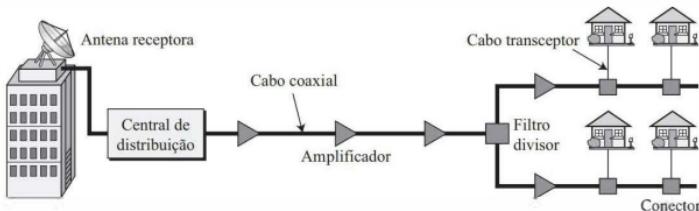



Figura 5.63 Rede de TV a cabo tradicional.

O escritório da TV a cabo, conhecido como a *central de distribuição*, recebe os sinais de vídeo das estações de radiodifusão e alimenta os sinais em cabos coaxiais. Os sinais ficam cada vez mais fracos com a distância, de modo que foram instalados amplificadores ao longo de toda a rede para renová-los. Poderia haver até 35 amplificadores entre a central de distribuição e as instalações do assinante. Na outra extremidade, filtros divisores repartem o cabo, enquanto conectores e cabos transceptores criam as conexões com as instalações do assinante.

O sistema de TV a cabo tradicional utilizava cabos coaxiais fim a fim. Devido à atenuação dos sinais e à utilização de um grande número de amplificadores, a comunicação na rede tradicional era unidirecional (apenas ida). Os sinais de vídeo eram transmitidos da central de distribuição para as instalações do assinante.

Rede Híbrida Fibra-Coaxial

A segunda geração de redes a cabo é chamada **rede Híbrida Fibra-Coaxial** (HFC – Hybrid Fiber--Coaxial). A rede usa uma combinação de fibra óptica e cabo coaxial. O meio de transmissão do escritório de TV a cabo até uma caixa, conhecida como *nó óptico*, é uma fibra óptica; do nó óptico até o interior das residências, passando pela vizinhança dos assinantes, o meio ainda é um cabo coaxial. A Figura 5.64 mostra um diagrama esquemático de uma rede HFC.

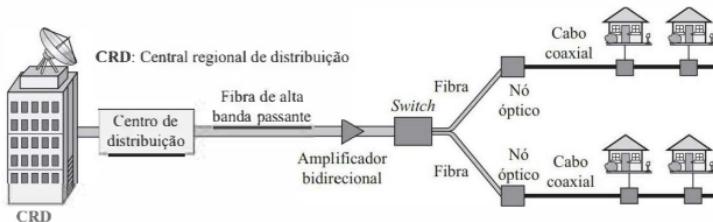



Figura 5.64 Rede Híbrida Fibra-Coaxial (HFC).

A *Central Regional de Distribuição* (CRD) normalmente serve até 400.000 assinantes. Os CRDs alimentam os *centros de distribuição*, cada qual servindo até 40.000 assinantes. O centro de distribuição desempenha um papel fundamental nessa nova infraestrutura. A modulação e a distribuição dos sinais são feitas aqui; os sinais são, então, enviados aos nós ópticos por meio de cabos de fibra óptica. O nó óptico divide os sinais analógicos de modo que o mesmo sinal é enviado para todos os cabos coaxiais. Cada cabo coaxial serve até 1.000 assinantes. A utilização de cabos de fibra óptica reduz a necessidade de amplificadores para oito ou menos.

Uma razão para migrar da infraestrutura tradicional para a híbrida é tornar a rede a cabo bidirecional (os dados trafegam nos dois sentidos).

TV a cabo para transferência de dados

As companhias de TV a cabo estão agora competindo com empresas de telefonia pelos clientes residenciais que desejam alta velocidade de transferência de dados. A tecnologia DSL oferece conexões com altas taxas de transferência de dados para assinantes residenciais por meio de um lacete local. No entanto, a tecnologia DSL utiliza os já existentes cabos de par trançado sem revestimento, os quais são muito suscetíveis a interferências. Isto impõe um limite superior à taxa de transferência de dados. Uma solução é a utilização da rede de TV a cabo. Nesta seção, discutimos brevemente essa tecnologia.

Mesmo em um sistema de HFC, a última parte da rede, do nó óptico até as instalações do assinante, ainda é formada por um cabo coaxial. Esse cabo coaxial tem uma largura de banda que varia de 5 a 750 MHz (aproximadamente). Para fornecer acesso à Internet, as empresas de TV a cabo dividiram essa largura de banda em três faixas: vídeo, recepção de dados e envio de dados, conforme mostra a Figura 5.65.

Envio de dados | Banda de vídeo | Recepção de dados

Frequência, MHz 5 42 54 550 750

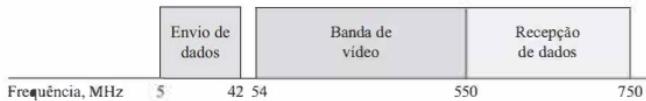



Figura 5.65 Divisão de banda pela CATV no cabo coaxial.

A *banda de vídeo* ocupa as frequências de 54 a 550 MHz. Como cada canal de TV ocupa 6 MHz, esta banda pode acomodar mais de 80 canais. A recepção de dados (vindos da Internet para as instalações do assinante) ocupa a banda superior, de 550 a 750 MHz. Essa banda também é dividida em canais de 6 MHz.

O envio de dados (das instalações do assinante para a Internet) ocupa a banda inferior, de 5 a 42 MHz. Essa banda também é dividida em canais de 6 MHz. Existem 2 *bits/baud* na modulação QPSK (*Quadrature Phase Shift Keying* ou Deslocamento de Fase em Quadratura). A norma especifica 1 Hz por cada *baud**, o que significa que, teoricamente, os dados podem ser enviados a 12 Mbps (2 *bits*/Hz × 6 MHz). No entanto, a taxa de dados é normalmente inferior a 12 Mbps.

Compartilhamento

Tanto a banda de envio como a de recepção são compartilhadas pelos assinantes. A largura de banda de dados de envio é de 37 MHz. Isto significa que existem apenas seis canais de 6 MHz disponíveis na direção de envio. Um assinante deve utilizar um canal para enviar seus dados para a Internet. A pergunta é: "Como seis canais podem ser compartilhados em uma área com 1.000, 2.000 ou até 100.000 assinantes?" A solução é o compartilhamento de tempo. A banda é dividida em canais; estes devem ser compartilhados entre os assinantes na mesma região. O provedor de serviços a cabo aloca um canal, estática ou dinamicamente, para um grupo de assinantes. Se um assinante desejar enviar dados, ele disputa o canal com os outros usuários que querem acessar a rede; o assinante deve esperar até que o canal esteja disponível.

A situação é semelhante na direção de recepção. A banda de recepção tem 33 canais de 6 MHz. Um provedor de serviços a cabo provavelmente tem mais de 33 assinantes; portanto, cada canal deve ser compartilhado entre um grupo de assinantes. No entanto, a situação é diferente na direção de recepção; nesse caso, temos um cenário de *multicast*. Se houver dados para qualquer um dos assinantes no grupo, eles são enviados para aquele canal. Os dados são enviados para todos os assinantes. Porém, como cada assinante também tem um endereço registrado junto ao provedor, o modem a cabo usado pelo grupo compara o endereço referente aos dados com o endereço atribuído pelo provedor. Se os endereços forem iguais, os dados são conservados; caso contrário, são descartados.

CM e CMTS

Para usar uma rede a cabo na transmissão de dados, precisamos de dois dispositivos essenciais: um **Modem a Cabo** (CM – Cable Modem) e um **Sistema de Transmissão por Modem a Cabo** (CMTS – Cable Modem Transmission System). O modem a cabo é colocado nas instalações do assinante. O **sistema de transmissão por modem a cabo é** instalado dentro da empresa de TV a cabo. O CMTS recebe dados da Internet e os envia para o assinante. Ele também recebe os dados do assinante e os passa para a Internet. Ele atua de forma semelhante a um modem ADSL. A Figura 5.66 mostra a localização desses dois dispositivos. Assim como na tecnologia DSL, a empresa de TV a cabo precisa se tornar um ISP e fornecer serviços de Internet para o assinante. Nas instalações do assinante, o CM separa o vídeo dos dados e os envia para o aparelho de televisão ou para o computador.

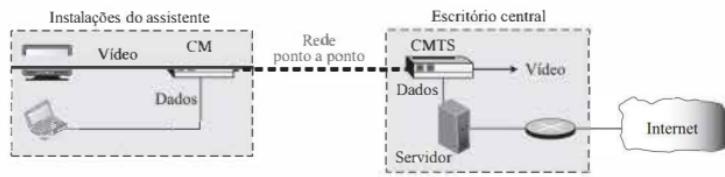



Figura 5.66 Sistema de Transmissão por Modem a Cabo (CMTS).

* N. de T.: O termo *baud* refere-se ao número de elementos de sinal transmitidos por segundo. Ver Capítulo 7 para obter mais detalhes.

5.6.2 SONET

Nesta seção, apresentamos uma rede de alta velocidade, a SONET, usada como uma rede de transporte para levar dados provenientes de outras redes. Primeiramente, discutimos a SONET como um protocolo, e então mostramos como as redes SONET podem ser construídas a partir dos padrões definidos no protocolo.

As elevadas larguras de banda do cabo de fibra óptica são adequadas para as tecnologias atuais que exigem alta taxa de transferência de dados (como videoconferência) e para o transporte simultâneo de dados pertencentes a um grande número tecnologias que exigem uma baixa taxa de transferência. Por isso, a importância das fibras ópticas cresce juntamente com o desenvolvimento de tecnologias que requerem alta taxa de transferência de dados ou ampla largura de banda para a transmissão. Com sua importância, veio a necessidade de padronização. Os Estados Unidos (ANSI) e a Europa (ITU-T) responderam a essa demanda com a especificação de normas que, embora independentes, são fundamentalmente semelhantes e compatíveis. O padrão ANSI é conhecido como **Rede Óptica Síncrona** (SONET – Synchronous Optical Network), e é discutido nesta seção.

Arquitetura

O SONET é uma rede síncrona que usa multiplexação síncrona por divisão de tempo TDM (Time Division Multiplexing). Todos os relógios do sistema são atrelados a um relógio-mestre. Primeiramente, introduzimos a arquitetura de um sistema SONET: sinais, dispositivos e conexões.

Sinais

O SONET define uma hierarquia de níveis de sinalização elétrica denominados **Sinais de Transporte Síncrono** (STSs – Synchronous Transport Signals). Cada nível STS (STS-1 a STS-192) suporta uma determinada taxa de transferência de dados, especificada em *megabits* por segundo (ver Tabela 5.10). Os sinais ópticos correspondentes são conhecidos como **Portadoras Ópticas** (OCs – Optical Carriers).

Tabela 5.10 Taxas de transferência no SONET.

STS	OC	Taxa (Mbps)	STS	OC	Taxa (Mbps)
STS-1	OC-1	51.840	STS-24	OC-24	1244.160
STS-3	OC-3	155.520	STS-36	OC-36	1866.230
STS-9	OC-9	466.560	STS-48	OC-48	2488.320
STS-12	OC-12	622.080	STS-96	OC-96	4976.640
STS-18	OC-18	933.120	STS-192	OC-192	9953.280

Observando a Tabela 5.10, percebemos alguns pontos interessantes. Em primeiro lugar, o nível mais baixo nessa hierarquia apresenta uma taxa de transferência de dados de 51.840 Mbps, que é maior do que aquela do serviço DS-3 (44.736 Mbps, explicado no Capítulo 7). Na realidade, o STS-1 foi projetado para acomodar taxas de dados equivalentes às do DS-3. A diferença de capacidade é fornecida para que seja possível tratar a carga adicional introduzida pelo sistema óptico. Em segundo lugar, a taxa de transferência do STS-3 é exatamente três vezes a taxa do STS-1, enquanto a taxa do STS-9 é exatamente metade da taxa do STS-18. Essas relações indicam que 18 canais STS-1 podem ser multiplexados em um canal STS-18, seis canais STS-3 podem ser multiplexados em um canal STS-18 e assim por diante.

Dispositivos SONET

A Figura 5.67 mostra uma conexão simples usando dispositivos SONET. A transmissão via SONET depende de três dispositivos básicos: multiplexadores/demultiplexadores STS, regeneradores, multiplexadores de inserção/remoção (ADM – Add/Drop Multiplexers) e terminais.

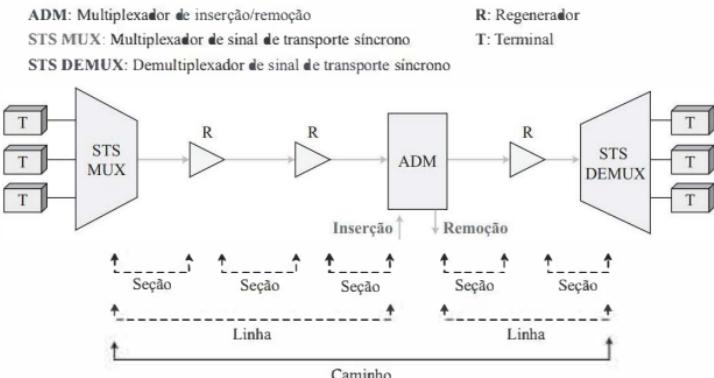



Figura 5.67 Uma rede simples usando dispositivos SONET.

Multiplexador/demultiplexador STS Multiplexadores/demultiplexadores STS marcam os pontos de início e fim de um enlace SONET. Eles fornecem a interface entre um ramo de uma rede elétrica e a rede óptica. Um *multiplexador STS* multiplexa sinais provenientes de várias fontes elétricas e cria o sinal OC correspondente. Um *demultiplexador STS* demultiplexa um sinal óptico OC nos sinais elétricos correspondentes.

Regenerador Regeneradores estendem o comprimento das conexões. Um *regenerador* é um repetidor (discutido mais adiante) que leva um sinal óptico recebido (OC-*n*), faz a sua demodulação para transformá-lo no sinal elétrico correspondente (STS-*n*), regenera o sinal elétrico e finalmente modula o sinal elétrico em seu sinal OC-*n* correspondente. Um regenerador SONET substitui algumas das informações adicionais existentes (informações de cabeçalho) por novas.

Multiplexador de inserção/remoção Multiplexadores de inserção/remoção permitem a inserção e extração de sinais. Um *Multiplexador de inserção/remoção* (ADM – Add/Drop Multiplexer) pode inserir STSs provenientes de diferentes origens em um determinado caminho, ou pode remover um sinal específico de um caminho e redirecioná-lo sem demultiplexar o sinal todo. Em vez de depender de temporização e posições de *bits*, os multiplexadores de inserção/remoção usam informações de cabeçalho, como endereços e ponteiros (descritos mais adiante nesta seção) para identificar os fluxos individuais.

Terminais Um *terminal* é um dispositivo que usa os serviços de uma rede SONET. Por exemplo, na Internet, um terminal pode ser um roteador que precisa enviar pacotes para outro roteador na outra extremidade de uma rede SONET.

Conexões

Os dispositivos definidos na seção anterior são conectados usando *seções*, *linhas* e *caminhos*.

Seções

Uma *seção* é um enlace óptico conectando dois dispositivos vizinhos: multiplexador a multiplexador, multiplexador a regenerador, ou regenerador a regenerador.

Linhas

Uma *linha* refere-se à região da rede entre dois multiplexadores.

Caminhos

Um *caminho* é uma região fim a fim da rede entre dois multiplexadores STS. Em uma SONET simples contendo dois multiplexadores STS conectados diretamente um ao outro, a seção, a linha e o caminho são equivalentes.

Camadas SONET

O padrão SONET inclui quatro camadas funcionais: a camada fotônica, a de seção, a de linha e a de caminho. Elas correspondem às camadas física e de enlace de dados. Os cabeçalhos adicionados ao quadro nas várias camadas são discutidos mais adiante neste capítulo. A Figura 5.68 mostra as camadas e a relação entre as camadas e os dispositivos descritos anteriormente.

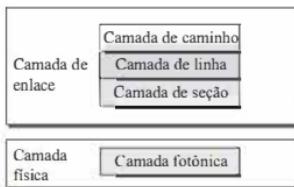



Figura 5.68 Camadas SONET comparadas com as camadas OSI e Internet.

Camada de caminho

A *camada de caminho* é responsável pela movimentação de um sinal de sua origem óptica para o seu destino óptico. Na origem óptica, o sinal é transformado de sua forma eletrônica em sua forma óptica, multiplexado com outros sinais e encapsulado em um quadro. No destino óptico, o quadro é recebido e demultiplexado, sendo os sinais ópticos individuais transformados de volta em suas formas eletrônicas. As informações relativas à camada de caminho são adicionadas nessa camada. Multiplexadores STS fornecem funções da camada de caminho.

Camada de linha

A *camada de linha* é responsável pela movimentação de um sinal ao longo de uma linha física. As informações relativas à camada de linha são adicionadas ao quadro nessa camada. Os multiplexadores STS e os multiplexadores de inserção/remoção fornecem funções da camada de linha.

Camada de seção

A *camada de seção* é responsável pela movimentação de um sinal ao longo de uma seção física. Ela lida com enquadramento, embaralhamento e controle de erros. As informações relativas à camada de seção são adicionadas ao quadro nessa camada.

Camada fotônica

A *camada fotônica* corresponde à camada física. Ela inclui especificações físicas para o canal de fibra óptica, sensibilidade do receptor, funções de multiplexação, e assim por diante. O SONET usa a codificação NRZ (ver o Capítulo 7) na qual a presença de luz representa 1 e a ausência de luz representa 0.

Quadros SONET

Cada sinal de transporte síncrono STS-*n* é composto por 8.000 quadros. Cada quadro consiste em uma matriz bidimensional de *bytes* com 9 linhas por 90 × *n* colunas. Por exemplo, um quadro STS-1 apresenta 9 linhas por 90 colunas (810 *bytes*), enquanto um quadro STS-3 consiste em 9 linhas por 270 colunas (2.430 *bytes*). A Figura 5.69 mostra o formato geral de um STS-1 e de um STS-*n*.

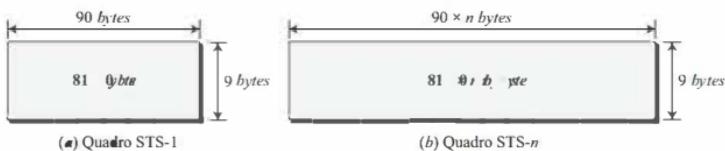



Figura 5.69 Um quadro STS-1 e um quadro STS-*n*.

Transmissão de quadros, *bytes* e *bits*

Um dos pontos interessantes do SONET é que cada sinal STS-*n* é transmitido a uma taxa fixa de 8.000 quadros por segundo. Essa é a taxa na qual a voz é digitalizada (ver o Capítulo 7). Para cada quadro, os *bytes* são transmitidos da esquerda para a direita, de cima para baixo. Para cada *byte*, os *bits* são transmitidos do mais significativo para o menos significativo (esquerda para a direita). A Figura 5.70 mostra a ordem de transmissão de quadros e *bytes*.

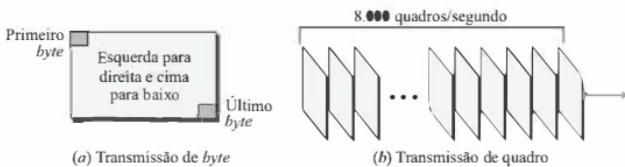



Figura 5.70 Quadros STS-1 em trânsito.

Um sinal SONET STS-*n* é transmitido a uma taxa de 8.000 quadros por segundo.

Se amostrarmos um sinal de voz e utilizarmos 8 *bits* (1 *byte*) para representar cada amostra, podemos dizer que cada *byte* em um quadro SONET pode transportar informações vindas de um canal de voz digitalizado. Em outras palavras, um sinal STS-1 pode transportar 774 canais de voz simultaneamente (810 menos 36 *bytes* necessários para informações de controle adicionadas pelas camadas).

Cada *byte* em um quadro SONET pode transportar um canal de voz digitalizada.

Perceba que, no SONET, existe uma relação exata entre as taxas de transferência de dados de diferentes sinais STS. Podemos calcular a taxa de transferência de dados do STS-3 usando o taxa de transferência de dados do STS-1 (multiplicando o valor deste último por 3).

No SONET, a taxa de transferência de dados de um sinal STS-*n* é *n* vezes o valor da taxa de transferência de dados de um sinal STS-1.

Formato dos quadros STS-1

O formato básico de um quadro STS-1 é mostrado na Figura 5.71. Como dissemos anteriormente, um quadro SONET é uma matriz de 9 linhas de 90 *bytes* (octetos) cada, totalizando 810 *bytes*.

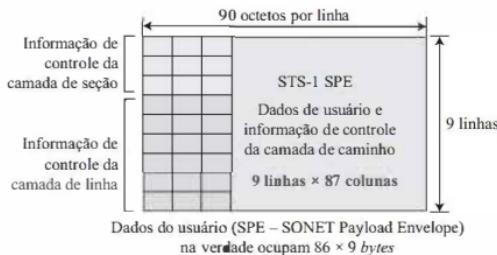



Figura 5.71 Informação de controle no quadro STS-1.

Multiplexação STS

No SONET, quadros transmitidos a uma taxa menor podem ser sincronamente multiplexados usando divisão de tempo para gerar um quadro com maior taxa de transferência. Por exemplo, três sinais (canais) STS-1 podem ser combinados em um sinal (canal) STS-3, quatro STS-3 podem ser multiplexados em um STS-12, e assim por diante.

A multiplexação é do tipo TDM síncrono, e todos os relógios na rede são atrelados a um relógio-mestre para conseguir a sincronização.

Em SONET, todos os relógios na rede são atrelados a um relógio-mestre.

Precisamos ressaltar que a multiplexação também pode acontecer com taxas de transferência de dados maiores. Por exemplo, quatro sinais STS-3 podem ser multiplexados em um sinal STS-12, conforme mencionado anteriormente. No entanto, os sinais STS-3 precisam primeiro ser demultiplexados em 12 sinais STS-1, e em seguida esses doze sinais precisam ser multiplexados em um sinal STS-12. A razão para esse trabalho extra ficará mais clara depois que discutirmos o mecanismo de intercalação de *bytes*.

Multiplexador de inserção/remoção

A multiplexação de vários sinais STS-1 em um sinal STS-*n* é feita no multiplexador STS (na camada de caminho). A demultiplexação de um sinal STS-*n* em componentes STS-1 é feita no demultiplexador STS. Entre esses dois dispositivos, no entanto, o SONET usa multiplexadores de inserção/remoção que podem substituir um sinal por outro. Precisamos deixar claro que essa não é uma demultiplexação/multiplexação no sentido convencional. Um multiplexador de inserção/remoção opera na camada de linha. Um multiplexador de inserção/remoção não adiciona informações de controle das camadas de seção, linha ou caminho. Ele funciona quase como um *switch*; remove um sinal STS-1 e insere outro. O tipo de sinal na entrada e saída de um multiplexador de inserção/remoção permanece o mesmo (ambos STS-3 ou ambos STS-12, por exemplo). O multiplexador de inserção/remoção (ADM), apenas remove os *bytes* correspondentes e os substitui pelos novos *bytes* (incluindo os *bytes* de controle das camadas de seção e de linha).

Redes SONET

Usando equipamentos SONET, podemos criar uma rede SONET que pode ser usada como um *backbone* de alta velocidade transportando dados provenientes de outras redes, como ATM ou IP. Podemos dividir as redes SONET basicamente em três categorias: redes lineares, em anel e em malha.

Rede linear

Uma rede linear é normalmente composta por multiplexadores e demultiplexadores STS, alguns geradores e alguns multiplexadores de inserção/remoção, conforme mostra a Figura 5.72.

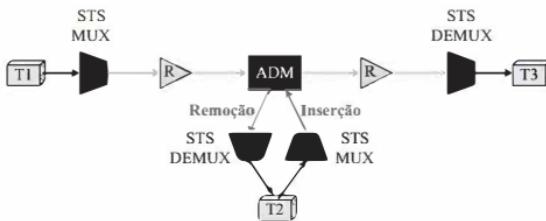



Figura 5.72 Uma rede SONET linear.

Redes em anel

ADMs permitem a criação de redes SONET em anel. Anéis SONET podem ser usados tanto em uma configuração unidirecional como bidirecional. Em cada caso, podemos acrescentar anéis extras para tornar a rede autorrecuperável, ou seja, capaz de se recuperar automaticamente de uma falha na linha. A Figura 5.73 mostra uma rede em anel.

Embora tenhamos colocado um emissor e três receptores na figura, muitas outras configurações são possíveis. O emissor usa uma conexão bidirecional para enviar dados para ambos os anéis simultaneamente; o receptor usa *switches* de seleção para escolher o anel com uma melhor qualidade de sinal. Usamos um multiplexador STS e três demultiplexadores STS para enfatizar que os nós operam na camada de caminho.

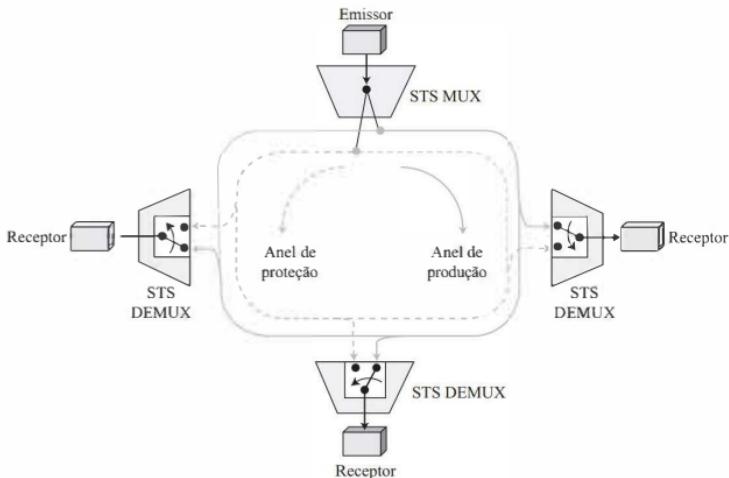



Figura 5.73 Um anel comutado unidirecional.

Redes em malha

Um problema com as redes em anel é a falta de escalabilidade. Quando o tráfego em um anel aumenta, precisamos atualizar não apenas as linhas, mas também os ADMs. Nessa situação, uma rede em malha com *switches* provavelmente apresentaria um melhor desempenho. Um *switch* em uma rede em malha é conhecido como *switch* de conexão cruzada. Um *switch* de conexão cruzada, assim como outros *switches* que discutimos anteriormente, tem portas de entrada e de saída. Em uma porta de entrada, o *switch* recebe um sinal OC-*n*, o transforma em um sinal STS-*n*, o demultiplexa nos sinais STS-1 correspondentes e envia cada sinal STS-1 para a porta de saída apropriada. Uma porta de saída recebe os sinais STS-1 provenientes de diferentes portas de entrada, os multiplexa em um sinal STS-*n*, e cria um sinal OC-*n* para a transmissão. A Figura 5.74 mostra uma rede SONET em malha e a estrutura de *switch*.

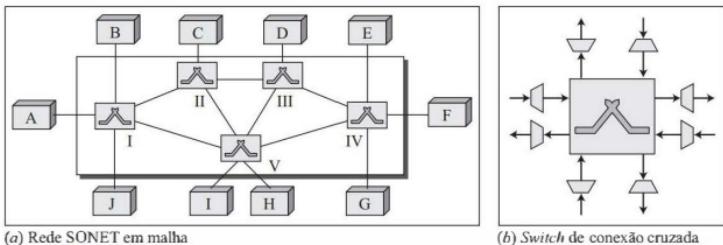



Figura 5.74 Uma rede SONET em malha.

Tributários virtuais

O SONET foi projetado para transportar dados em banda larga. Conforme discutido mais adiante, as taxas de transferência de dados atuais na hierarquia digital (DS-1 a DS-3) são, entretanto, inferiores às do STS-1. Para tornar o SONET compatível com a hierarquia atual, o projeto de seus quadros inclui um sistema de **Tributários Virtuais** (VTs – Virtual Tributaries), conforme mostra a Figura 5.75. Um tributário virtual é uma carga parcial de dados que pode ser inserida em uma conexão STS-1 e combinada com outras cargas parciais de dados para preencher o quadro. Em vez de usar todas as 86 colunas de carga útil de um quadro STS-1 para os dados de uma única origem, podemos subdividir a Carga Útil do Envelope SONET (SPE – SONET Payload Envelope) e chamar cada componente de um VT.

Figura 5.75 Tributários virtuais.

Foram definidos quatro tipos de VTs para acomodar hierarquias digitais existentes. Perceba que o número de colunas permitidas para cada tipo de VT pode ser determinado dobrando-se o número de identificação do tipo (o VT1.5 apresenta três colunas, o VT2 apresenta quatro colunas, etc.). O VT1.5 acomoda o serviço DS-1 dos Estados Unidos (1.544 Mbps). O VT2 acomoda o serviço europeu CEPT-1 (2.048 Mbps). O VT3 acomoda o serviço DS-1C (DS-1 fracionário, que opera a 3.152 Mbps). O VT6 acomoda o serviço DS-2 (6.312 Mbps).

Quando dois ou mais tributários são inseridos em um único quadro STS-1, eles são intercalados coluna a coluna. O SONET fornece mecanismos para identificar cada VT e separá-los sem demultiplexar o fluxo todo. A discussão sobre esses mecanismos e as questões de controle por trás deles está além do escopo deste livro.

5.6.3 Rede comutada: ATM

O **Modo de Transferência Assíncrona** (ATM – Asynchronous Transfer Mode) é um protocolo voltado a redes de longa distância comutadas e é baseado no protocolo de *comutação de células* projetado pelo Fórum ATM e adotado pelo ITU-T. A combinação de ATM com SONET permite a interconexão em alta velocidade de todas as redes do mundo. Na realidade, o ATM pode ser pensado como a "autoestrada" no termo "superautoestrada da informação"*.

O ATM utiliza multiplexação estatística (assíncrona) por divisão de tempo – razão pela qual ele é chamado de Modo de Transferência Assíncrona – para multiplexar células provenientes de diferentes canais. Ele usa faixas de tempo (*slots*) de tamanho fixo (o tamanho de uma célula). Os multiplexadores ATM preenchem uma faixa com uma célula proveniente de qualquer canal de entrada que tenha uma célula; a faixa permanece vazia se nenhum dos canais tiver uma célula para enviar. A Figura 5.76 mostra como as células provenientes de três entradas são multiplexadas. No primeiro sinal do relógio, o canal 2 não tem células (faixa de entrada vazia),

* N. de T.: O termo "superautoestrada da informação" (*information superhighway*) foi cunhado por Albert Gore na década de 1990, referindo-se aos sistemas de comunicação digital na Internet.

de modo que o multiplexador preenche a faixa com uma célula proveniente do terceiro canal. Após todas as células provenientes de todos os canais terem sido multiplexadas, as faixas de saída ficam vazias.

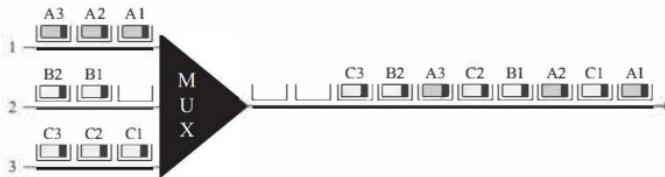



Figura 5.76 Multiplexação no ATM.

Arquitetura

Redes ATM são redes de comutação de células. Os dispositivos de acesso do usuário, denominados terminais ATM, são conectados por meio de uma **Interface Usuário-Rede** (UNI – User-to-Network Interface) aos *switches* dentro da rede. Os *switches* são conectados por meio de **Interfaces Rede-Rede** (NNIs – Network-to-Network Interfaces). A Figura 5.77 mostra um exemplo de uma rede ATM.

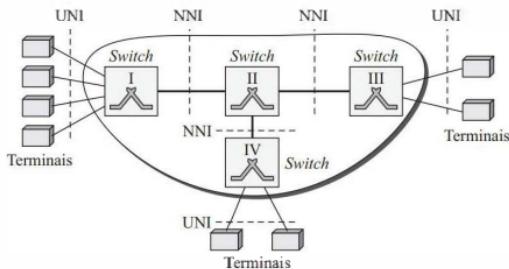



Figura 5.77 Arquitetura de uma rede ATM.

Conexão virtual

A conexão entre dois terminais é realizada por meio de caminhos de transmissão, caminhos virtuais e circuitos virtuais. Um **Caminho de Transmissão** (TP – Transmission Path) é uma conexão física (fios, cabo, satélite, e assim por diante) entre um terminal e um *switch* ou entre dois *switches*. Imagine os dois *switches* como duas cidades. Um caminho de transmissão pode consistir no conjunto de todas as rodovias que conectam diretamente as duas cidades.

Um caminho de transmissão é dividido em diversos caminhos virtuais. Um **Caminho Virtual** (VP – Virtual Path) fornece uma conexão ou um conjunto de conexões entre dois *switches*. Um caminho virtual pode ser imaginado como uma estrada que liga duas cidades. Cada estrada é um caminho virtual; o conjunto de todas as estradas é o caminho de transmissão.

Redes de células são baseadas em **Circuitos Virtuais** (VCs – Virtual Circuits). Todas as células pertencentes a uma única mensagem seguem o mesmo circuito virtual e conservam a sua ordenação

original até chegarem ao destino. O circuito virtual pode ser imaginado como as pistas de uma rodovia (caminho virtual). A Figura 5.78 mostra a relação entre um caminho de transmissão (uma ligação física), caminhos virtuais (uma combinação de circuitos virtuais que são agrupados porque partes de seus caminhos coincidem), e circuitos virtuais que conectam logicamente dois pontos.

Figura 5.78 TP, VPs e VCs.

Identificadores Em uma rede de circuitos virtuais, para que seja possível rotear os dados de um terminal até outro, as conexões virtuais precisam ser identificados. Com esse objetivo, os projetistas do ATM criaram um identificador hierárquico com dois níveis: um Identificador de Caminho Virtual (VPI – Virtual-Path Identifier) e um Identificador de Circuito Virtual (VCI – Virtual-Circuit Identifier). O VPI define o VP específico e o VCI define um VC em particular dentro do VP. O VPI é o mesmo para todas as conexões virtuais agrupadas (logicamente) em um VP.

Os comprimentos dos VPIs para as UNIs e para as NNIS são diferentes. Em uma UNI, o VPI tem 8 *bits*, enquanto em uma NNI, o VPI tem 12 *bits*. O comprimento do VCI é o mesmo em ambas as interfaces (16 *bits*). Podemos, portanto, dizer que uma conexão virtual é identificada por 24 *bits* em uma UNI e por 28 *bits* em uma NNI, conforme mostra a Figura 5.79.

A ideia por trás de dividir um identificador de circuito virtual em duas partes é permitir o roteamento hierárquico. A maioria dos *switches* em uma rede ATM típica faz o roteamento usando VPIs. Os *switches* na borda da rede, que interagem diretamente com os dispositivos terminais, usam tanto os VPIs como os VCIs.

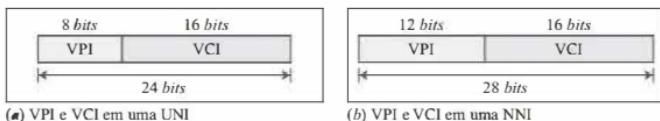



Figura 5.79 Identificadores de conexão virtual em UNIs e NNIs.

Células A unidade básica de dados em uma rede ATM é denominada uma célula. Uma célula tem apenas 53 *bytes* de comprimento, com 5 *bytes* atribuídos ao cabeçalho e 48 *bytes* transportando a carga útil (os dados do usuário podem ser menores do que 48 *bytes*). A maior parte do cabeçalho é ocupada pelo VPI e pelo VCI, que definem a conexão virtual pela qual uma célula deve viajar de um terminal até um *switch* ou de um *switch* até outro. A Figura 5.80 mostra a estrutura das células.

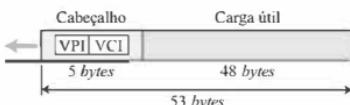



Figura 5.80 Uma célula ATM.

Estabelecimento e encerramento de conexões

O ATM usa dois tipos de conexões: PVC e SVC.

PVC No Circuito Virtual Permanente (PVC – Permanent Virtual-Circuit) é estabelecida entre dois terminais pelo provedor da rede. Os VPIs e os VCIs são definidos para as conexões permanentes, e os valores são inseridos nas tabelas de cada *switch*.

SVC No Circuito Virtual Comutado (SVC – Switched Virtual-Circuit), cada vez que um terminal deseja se conectar a outro, um novo circuito virtual deve ser estabelecido. O ATM não pode realizar essa tarefa sozinho. Ele precisa dos endereços da camada de rede e dos serviços de outro protocolo (tal como o IP). O mecanismo de sinalização desse outro protocolo cria uma solicitação de conexão usando os endereços da camada de rede dos dois terminais. O mecanismo exato depende do protocolo da camada de rede.

Comutação

O ATM usa *switches* para rotear as células de um terminal de origem ao terminal de destino. Um *switch* encaminha a célula usando tanto os VPIs como os VCIs. O encaminhamento requer o identificador completo. A Figura 5.81 mostra como um *switch* de um PVC encaminha as células. Uma célula cujo VPI é 153 e cujo VCI é 67 chega à interface (porta) 1 do *switch*. O *switch* verifica sua tabela de comutação, que contém seis informações por linha: número da interface de chegada, VPI de entrada, VCI de entrada, número da interface de saída correspondente, o novo VPI e o novo VCI. O *switch* encontra a linha da tabela com interface 1, VPI 153 e VCI 67, e então descobre que essa combinação corresponde à interface de saída 3, com VPI 140 e VCI 92. Ele altera os valores de VPI e VCI no cabeçalho para 140 e 92, respectivamente, e envia a célula pela interface 3.

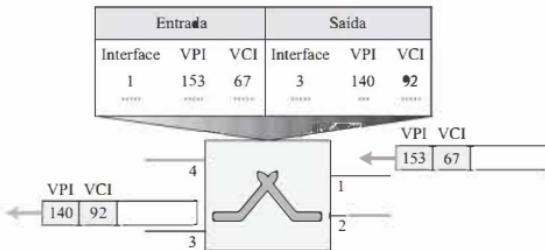



Figura 5.81 Roteamento com um *switch*.

Camadas ATM

O padrão ATM define três camadas. Elas são, de cima para baixo, a camada de adaptação ATM, a camada ATM e a camada física, conforme mostra a Figura 5.82. Os terminais usam as três camadas, enquanto os *switches* usam apenas as duas camadas inferiores.

Camada AAL A **Camada de Adaptação ATM** (AAL – ATM Adaptation Layer) foi projetada para dar suporte a dois conceitos do ATM. Primeiro, o ATM deve aceitar qualquer tipo de carga útil, seja ela um quadro de dados ou um fluxo de *bits*. Um quadro de dados pode vir de um protocolo da camada superior que cria quadros claramente delimitados para serem enviados para uma rede de transporte, tal como o ATM. Um bom exemplo é a Internet. O ATM também deve ser capaz de transportar dados multimídia. Ele pode aceitar fluxos contínuos de *bits* e quebrá-los em grupos para serem encapsulados em uma célula na camada ATM. A camada AAL usa duas subcamadas para realizar essas tarefas.

Figura 5.82 Camadas ATM.

Sejam os dados um quadro de dados ou um fluxo de *bits*, a carga útil deve ser dividida em segmentos de 48 *bytes* para serem transportados por uma célula. No destino, esses segmentos precisam ser remontados para recriar a carga útil original. A camada AAL define uma subcamada, conhecida como subcamada **Segmentação e Remontagem** (SAR – Segmentation And Reassembly), para fazer isso. A segmentação é feita na origem, enquanto a remontagem acontece no destino.

Antes de os dados serem segmentados pela subcamada SAR, eles devem ser preparados para garantir sua integridade. Isso é feito por uma subcamada denominada **Subcamada de Convergência** (CS – Convergence Sublayer). O ATM define quatro versões da camada AAL: *AAL1*, *AAL2*, *AAL3/4* e *AAL5*. Discutimos aqui apenas a AAL5, que é utilizada na Internet atual.

A subcamada ALL5 foi projetada para aplicações da Internet. Ela também é conhecida como **Camada de Adaptação Simples e Eficiente** (SEAL – Simple and Efficient Adaptation Layer). A AAL5 assume que todas as células que pertencem a uma única mensagem viajam sequencialmente e que as funções de controle são incluídas nas camadas superiores do aplicativo que está enviando os dados. A Figura 5.83 mostra a subcamada AAL5.

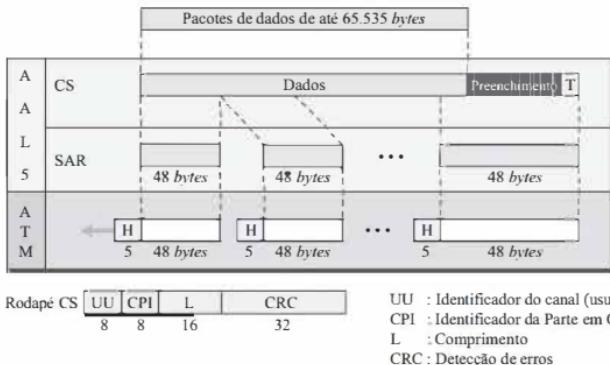



Figura 5.83 AAL5

O pacote na camada CS usa um rodapé com quatro campos. O campo UU é o identificador usuário a usuário. O IPC é o identificador da parte em comum. O campo C define o comprimento dos dados originais. O campo CRC é um campo de dois *bytes* usado para verificação de erros nos dados completos.

Camada ATM A *camada ATM* fornece serviços de roteamento, gerenciamento de tráfego, comutação e multiplexação. Ela processa o tráfego de saída, aceitando segmentos de 48 *bytes* das subcamadas AAL e transformando-os em células de 53 *bytes* por meio da adição de um cabeçalho de 5 *bytes*.

Camada física Como na Ethernet e nas LANs sem fio, as células ATM podem ser transportadas por qualquer tipo de camada física.

Controle de congestionamento e qualidade de serviço

O ATM inclui mecanismos de controle de congestionamento e qualidade de serviço bastante avançados.

5.7 DISPOSITIVOS DE CONEXÃO

Estações e redes normalmente não operam isoladamente. Usamos dispositivos de conexão para interligar estações de modo a criar uma rede ou para interligar redes com o objetivo de criar uma internet. Os dispositivos de conexão podem operar em diferentes camadas do modelo Internet. Discutiremos três tipos de dispositivos de conexão: repetidores e *hubs*, *switches* da camada de enlace (ou comutadores da camada 2) e roteadores (ou comutadores da camada 3). Os repetidores e os *hubs* operam na primeira camada do modelo Internet. *Switches* da camada de enlace operam nas duas primeiras camadas. Roteadores operam nas três primeiras camadas.

5.7.1 Repetidores e *hubs*

Um **repetidor** é um dispositivo que opera apenas na camada física. Os sinais que transportam informações dentro de uma rede são capazes de percorrer uma distância fixa antes que a atenuação ponha em perigo a integridade dos dados. Um repetidor recebe um sinal e, antes que ele fique muito fraco, ou seja, corrompido, tal dispositivo *regenera* e *retemporiza* a sequência original de *bits*. Em seguida, o repetidor envia o sinal restaurado. No passado, quando LANs Ethernet usavam uma topologia em barramento, repetidores eram usados para conectar dois segmentos de uma LAN, de modo a superar as restrições de comprimento do cabo coaxial. Atualmente, entretanto, LANs Ethernet usam topologias em estrela. Em uma topologia em estrela, um repetidor é um dispositivo de múltiplas portas, geralmente conhecido como *hub*, que pode servir como ponto de conexão e, ao mesmo tempo, funcionar como um repetidor. A Figura 5.84 mostra que, quando um pacote que vai da estação A para a estação B chega ao *hub*, o sinal que representa o quadro é regenerado para remover qualquer ruído que possa corromper os dados, mas o *hub* encaminha o pacote por todas as portas de saída, exceto por aquela na qual o sinal foi recebido. Em outras palavras, o quadro é transmitido via *broadcast*. Todas as estações na LAN recebem o quadro, mas apenas a estação B o conserva. As outras estações o descartam. A Figura 5.84 mostra o papel de um *hub* em uma LAN comutada.

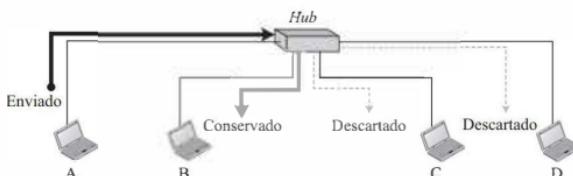



Figura 5.84 Repetidor ou *hub*.

A figura mostra definitivamente que um *hub* não apresenta qualquer capacidade de filtragem; ele não tem a inteligência necessária para determinar a porta pela qual o quadro deve sair.

Um *hub* não apresenta funcionalidades de filtragem.

Hubs e repetidores são dispositivos da camada física. Eles não apresentam um endereço da camada de enlace e não verificam o endereço da camada de enlace presente no quadro recebido. Apenas regeneram os *bits* corrompidos e enviam os quadros pelas portas de saída.

5.7.2 *Switches*

Um *switch da camada de enlace*, ou simplesmente *switch*, opera nas camadas física e de enlace de dados. Como um dispositivo da camada física, ele regenera o sinal recebido. Como um dispositivo da camada de enlace, ele pode verificar os endereços MAC (de origem e de destino) contidos no quadro.

Filtragem

Podemos nos perguntar qual é a diferença em termos de funcionalidade entre um *switch* e um *hub*. Um *switch* possui capacidade de *filtragem*; pode verificar o endereço de destino de um quadro e decidir por qual porta de saída ele deve ser enviado.

Um *switch* possui uma tabela usada em decisões de filtragem.

Vamos dar um exemplo. Na Figura 5.85, temos uma LAN com quatro estações conectadas a um *switch*. Se um quadro destinado à estação de endereço 71:2B:13:45:61:42 chegar à porta 1 do *switch*, ele consulta sua tabela para determinar a porta de saída.

Endereço	Porta
71:2B:13:45:61:41	1
71:2B:13:45:61:42	2
64:2B:13:45:61:12	3
64:2B:13:45:61:13	4

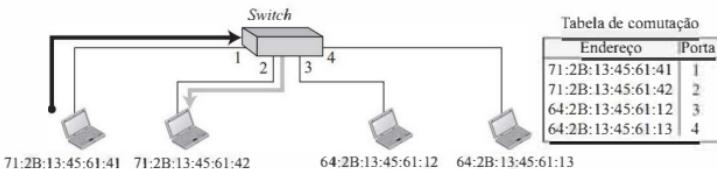



Figura 5.85 *Switch*

De acordo com a tabela, os quadros destinados a 71:2B:13:45:61:42 devem ser enviados somente pela porta 2 e, portanto, não há necessidade de transmitir o quadro às outras portas.

Um *switch* não altera os endereços da camada de enlace (MAC) em um quadro.

Switches transparentes

Um *switch transparente* é um *switch* cujas estações a ele conectadas não tomam ciência de sua existência. Se um *switch* é adicionado ou removido do sistema, não é necessário reconfigurar as estações. De acordo com a especificação do IEEE 8021d, um sistema equipado com *switches* transparentes deve satisfazer três requisitos:

- Os quadros devem ser encaminhados de uma estação para outra.
- A tabela de encaminhamento é criada automaticamente usando informações sobre a movimentação de quadro na rede (aprendizado).
- Laços devem ser evitados no sistema.

Encaminhamento

Um *switch* transparente deve encaminhar corretamente os quadros, conforme discutido na seção anterior.

Aprendizado

Os primeiros *switches* inventados apresentavam tabelas de encaminhamento estáticas. O administrador do sistema precisava inserir manualmente cada entrada na tabela durante a configuração do *switch*. Embora o processo fosse simples, ele não era prático. Se uma estação fosse adicionada ou removida, a tabela precisava ser modificada manualmente. O mesmo acontecia se o endereço MAC de uma estação mudasse, o que não é um evento raro. Por exemplo, colocar uma nova interface de rede significa ter um novo endereço MAC.

Uma solução mais adequada é substituir a tabela estática por uma tabela dinâmica que mapeia endereços das portas (interfaces) automaticamente. Para criar uma tabela dinâmica, precisamos de um *switch* que aprenda gradualmente com as informações de movimentação dos quadros. Para isso, o *switch* inspeciona os endereços de destino e de origem do quadro. O endereço de destino é usado para a decisão de encaminhamento (busca na tabela); já o endereço de origem é usado para adicionar entradas na tabela e para fins de atualização. Detalharemos esse processo usando a Figura 5.86.

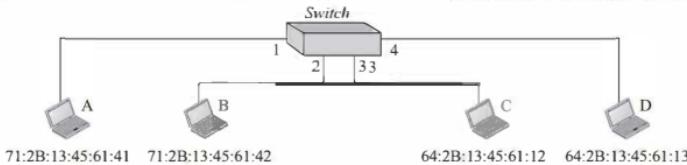



Figura 5.86 Aprendizado no *switch*.

1. Quando a estação A envia um quadro para a estação D, o *switch* não possui entradas nem para D nem para A. O quadro é enviado por todas as três portas, ou seja, inunda a rede. Entretanto, verificando o endereço de origem, o *switch* aprende que a estação A deve estar conectada à porta 1. Isto significa que quadros destinados a A, no futuro, devem sair pela porta 1. O *switch* adiciona esta entrada à sua tabela, que agora tem a sua primeira entrada.
2. Quando a estação D envia um quadro para a estação B, o *switch* não tem uma entrada para B, de modo que ele inunda a rede novamente. No entanto, adiciona uma entrada a mais à tabela, relacionada à estação D.
3. O processo de aprendizagem continua até que a tabela apresente informações sobre todas as portas.

No entanto, note que o processo de aprendizagem pode levar um longo tempo. Por exemplo, se uma estação não enviar quadros (uma situação rara), a estação nunca terá uma entrada na tabela.

5.7.3 Roteadores

Discutimos sobre roteadores no Capítulo 4. Neste capítulo, mencionamos os roteadores com o objetivo de compará-los com um *switch* e um *hub*. Um *roteador* é um dispositivo com três camadas; ele opera nas camadas: física, de enlace de dados e de rede. Como um dispositivo de camada física, ele regenera o sinal que recebe. Como um dispositivo da camada de enlace, ele verifica os endereços físicos (de origem e de destino) contidos no pacote. Como um dispositivo da camada de rede, um roteador verifica os endereços da camada de rede.

Um roteador é um dispositivo de três camadas (física, de enlace de dados e de rede).

Um roteador pode conectar redes. Em outras palavras, um roteador é um dispositivo de interconexão; conecta redes independentes para criar uma internet. De acordo com essa definição, duas redes conectadas por um roteador constituem uma internet.

Existem três grandes diferenças entre um roteador e um *hub* ou um *switch*.

1. Um roteador tem um endereço físico e um endereço lógico (IP) para cada uma de suas interfaces.
2. Um roteador atua apenas sobre os pacotes nos quais o endereço da camada de enlace de destino corresponde ao endereço da interface na qual o pacote chega.
3. Um roteador altera o endereço da camada de enlace do pacote (tanto de origem como de destino), quando ele encaminha o pacote.

Usemos a Figura 5.87 como exemplo: considere uma organização que tem dois edifícios separados com uma LAN Ethernet Gigabit instalada em cada prédio. A organização usa *switches* em cada LAN. As duas LANs podem ser conectadas para formar uma grande LAN usando a tecnologia Ethernet 10-Gigabit, acelerando a conexão com a Ethernet e a conexão com o servidor da organização. Um roteador pode, então, conectar o sistema todo à Internet.

Um roteador, como vimos no Capítulo 4, alterará o endereço MAC que ele recebe porque os endereços MAC têm jurisdição local apenas.

Um roteador altera os endereços da camada de enlace de um pacote.

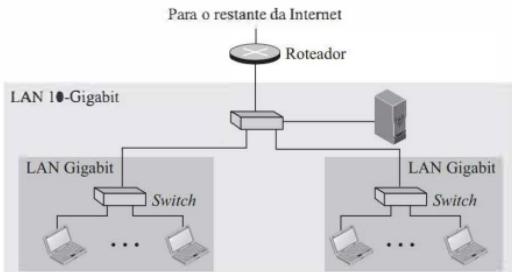



Figura 5.87 Exemplo de roteamento.

5.8 MATERIAL DO FINAL DO CAPÍTULO

5.8.1 Leitura recomendada

Para obter mais detalhes sobre os assuntos discutidos neste capítulo, recomendamos os seguintes livros. Os itens entre colchetes [...] referem-se à lista de referências no fim do texto.

Books

Diversos livros excelentes discutem as questões da camada de enlace. Entre eles, recomendamos a leitura de [Ham 80], [Zar 02], [Ror 96], [Tan 03], [GW 02], [For 03], [KMK 04], [Sta 04], [Kes 02], [PD 03], [Kei 02], [Spu 00], [KCK 98], [Sau 98], [Izz 00], [Per 00] e [WV 00].

RFCs

Uma discussão sobre o uso da soma de verificação na Internet pode ser encontrada na RFC 1141.

5.8.2 Termos-chave

- acesso aleatório
- acesso controlado
- Acesso Múltiplo com Detecção de Portadora (CSMA – Carrier Sense Multiple Access)
- Acesso Múltiplo com Detecção de Portadora / Detecção de Colisão (CSMA/CD – Carrier Sense Multiple Access/Collision Detection)
- Acesso Múltiplo com Detecção de Portadora / Prevenção de Colisão (CSMA/CA – Carrier Sense Multiple Access/Collision Avoidance)
- ALOHA
- ALOHA puro
- Antena Comunitária de TV (CATV – Community Antenna TV)
- Camada de Adaptação ATM (AAL – ATM Adaptation Layer)
- Camada de Adaptação Simples e Eficiente (SEAL – Simple and Efficient Adaptation Layer)
- Caminho de Transmissão (TP – Transmission Path)
- Caminho Virtual (VP – Virtual Path)
- canalização
- Circuito Virtual (VC – Virtual Circuit)
- código de paridade
- complemento de um
- contenção
- Controle de Acesso ao Meio (MAC – Media Access Control)

- Controle de Enlace de Dados (DLC – Data Link Control)
- Controle de Enlace de Dados de Alto Nível (HDLC – High-level Data Link Control)
- Controle de Enlace Lógico (LLC – Logical Link Control)
- demodulador
- distância de Hamming
- erro de um único *bit*
- erro em rajada
- Ethernet 10-Gigabit
- Ethernet Gigabit
- Ethernet Padrão
- Fast Ethernet
- *hub*
- Interface Rede-Rede (NNI – Network-to-Network Interface)
- Interface Usuário-Rede (UNI – User-to-Network Interface)
- Linha de Assinante Digital (DSL – Digital Subscriber Line)
- modem
- Modo de Transferência Assíncrona (ATM – Asynchronous Transfer Mode)
- Modo Balanceado Assíncrono (ABM – Asynchronous Balanced Mode)
- modulador
- Multiplexação por Divisão de Tempo (TDM – Time-Division Multiplexing)
- palavra de dados
- palavra de código
- Passagem de Ficha (*Token Passing*)
- Placa de Interface de Rede (NIC – Network Interface Card)
- Portadora Óptica (OC – Optical Carrier)
- preenchimento de *bits*
- preenchimento de *byte*
- Projeto 802
- Protocolo de Autenticação por Desafio-Resposta (CHAP – Challenge Handshake Authentication Protocol)
- Protocolo de Controle de Enlace (LCP – Link Control Protocol)
- Protocolo de Controle de Rede da Internet (IPCP – Internet Protocol Control Protocol)
- Protocolo Ponto a Ponto (PPP – Point-to-Point Protocol)
- Protocolo de Resolução de Endereços (ARP – Address Resolution Protocol)
- Rede Híbrida Fibra-Coaxial (HFC – Hybrid Fiber-Coaxial)
- Rede Local Virtual (VLAN – Virtual Local Area Network)
- Rede Óptica Síncrona (SONET – Synchronous Optical Network)
- Segmentação e Remontagem (SAR – Segmentation And Reassembly)
- Sinal de Transporte Síncrono (STS – Synchronous Transport Signal)
- síndrome
- *slotted* ALOHA
- Subcamada de Convergência (CS – Convergence Sublayer)
- *switch*
- Tributários Virtuais (VT – Virtual Tributaries)
- varredura (*polling*)
- Verificação de Redundância Cíclica (CRC – Cyclic Redundancy Check)

5.8.3 Resumo

Podemos considerar a camada de enlace de dados como sendo duas subcamadas. A subcamada superior é responsável pelo controle do enlace de dados, enquanto a inferior é responsável por resolver o acesso ao meio compartilhado. A subcamada de Controle de Enlace de Dados (DLC – Data Link Control) lida com o projeto e os procedimentos para a comunicação entre dois nós adjacentes: comunicação nó a nó. Essa subcamada é responsável por realizar o enquadramento e controlar erros. O controle de erros consiste em mecanismos para lidar com a corrupção dos dados durante a transmissão. Discutimos dois protocolos da camada de enlace neste capítulo: o HDLC e o PPP. O Controle de Enlace de Dados de Alto Nível (HDLC – High-level Data Link Control) é um protocolo orientado a *bits* usado para comunicação por meio de enlaces ponto a ponto e multiponto.

Entretanto, o protocolo mais comum para acesso ponto a ponto é o Protocolo Ponto a Ponto (PPP – Point-to-Point Protocol ou), que é um protocolo orientado a *bytes*.

Muitos protocolos formais foram criados para lidar com o acesso a um enlace compartilhado. Podemos categorizá-los em três grupos: protocolos de acesso aleatório, de acesso controlado e de canalização. Nos métodos de acesso aleatório ou de contenção, nenhuma estação tem prioridade ou controle sobre outra estação. No acesso controlado, as estações devem consultar umas às outras para descobrir qual estação tem direito de enviar dados. A canalização é um método de acesso múltiplo no qual a largura de banda disponível em um enlace é compartilhada no tempo, na frequência ou por meio de códigos, entre diferentes estações.

Na camada de enlace de dados, usamos endereçamento da camada de enlace. O sistema normalmente determina o endereço da camada de enlace do próximo nó usando o Protocolo de Resolução de Endereços (ARP – Address Resolution Protocol).

A Ethernet é o protocolo de rede local mais utilizado atualmente. A camada de enlace de dados da Ethernet consiste na subcamada LLC e na subcamada MAC. A subcamada MAC é responsável pela operação do método de acesso CSMA/CD e pelo enquadramento. Cada estação em uma rede Ethernet tem um endereço de 48 *bits* único impresso em sua placa de interface de rede (NIC – Network Interface Card). Uma Rede Local Virtual (VLAN – Virtual Local Area Network) é configurada via *software*, não por meio do cabeamento físico. A associação de máquinas a uma VLAN pode ser baseada em números de portas, endereços MAC, endereços IP, endereços IP *multicast*, ou uma combinação dessas características. VLANs são eficientes em termos de custo e tempo, podem reduzir o tráfego na rede, e podem ser usadas como uma medida adicional de segurança.

A necessidade de transmitir dados digitais levou à invenção do modem discado. Devido à necessidade de alta velocidade de recepção e envio de dados, as empresas de telefonia introduziram uma nova tecnologia, a *Linha de Assinante Digital* (DSL – Digital Subscriber Line). As redes a cabo foram originalmente criadas para proporcionar melhor acesso a programas de TV. As empresas de TV a cabo também descobriram um mercado promissor no fornecimento de acesso à Internet, usando alguns dos canais originalmente projetados para vídeo. O padrão Rede Óptica Síncrona (SONET – Synchronous Optical Network) foi desenvolvido pela ANSI para redes de fibra óptica. O Modo de Transferência Assíncrona (ATM – Asynchronous Transfer Mode) é um protocolo de comutação de células que, em combinação com a tecnologia SONET, permite conexões de alta velocidade. Uma célula é um pequeno bloco de informação de tamanho fixo. O pacote de dados do ATM é uma célula composta por 53 *bytes*. O padrão ATM define três camadas: a camada Camada de Adaptação ATM (AAL – ATM Adaptation Layer), a camada ATM e a camada física.

Um *hub* é um dispositivo de conexão que opera na camada física do modelo Internet. Um *switch* é um dispositivo de conexão que opera nas camadas físicas e de enlace de dados do modelo Internet. Um *switch* transparente pode encaminhar e filtrar quadros, construindo sua tabela de encaminhamento de forma automática. Um roteador é um dispositivo de conexão que opera nas três primeiras camadas da pilha de protocolos TCP/IP.

5.9 ATIVIDADES PRÁTICAS

5.9.1 Testes

Diversos testes interativos relativos a este capítulo podem ser encontrados no *site* www.mhhe.com/forouzan. Recomendamos que o estudante realize os testes para verificar o seu entendimento do material apresentado antes de continuar com as atividades práticas.

Questões

5-1 Descreva as diferenças entre a comunicação na camada de rede e a comunicação na camada de enlace de dados.

5-2 Descreva as diferenças entre um enlace ponto a ponto e um enlace de *broadcast*.

5-3 Explique por que é necessário utilizar marcadores quando usamos quadros de tamanho variável.

5-4 Explique por que não podemos aplicar preenchimento de *bits* no contexto de enquadramento orientado a caracteres para alterar um *byte* usado como marcador aparecendo no texto.

5-5 Quais as diferenças entre um erro de um único *bit* e um erro em rajada?

5-6 Qual é a definição de um código de bloco linear?

5-7 Em um código de bloco, uma palavra de dados tem 20 *bits* e a palavra de código correspondente tem 25 *bits*. Quais são os valores de k, r e n de acordo com as definições dadas no texto? Quantos *bits* redundantes são adicionados a cada palavra de dados?

5-8 Em uma palavra de código, acrescentamos dois *bits* redundantes em cada palavra de dados de oito *bits*. Determine o número de

a. palavras de código válidas.

b. palavras de código inválidas.

5-9 O que significa a distância de Hamming mínima no contexto de códigos corretores de erros?

5-10 Se quisermos detectar erros de dois *bits*, qual deve ser a distância de Hamming mínima?

5-11 A classe de código para detecção (e correção) de erros conhecida como código de Hamming consiste em códigos nos quais $d_{min} = 3$. Esse código pode detectar até dois erros (ou corrigir um único erro). Nele, os valores de n, k e r satisfazem as relações: $n = 2^r - 1$ e $k = n - r$. Encontre o número de *bits* nas palavras de dados e nas palavras de código se $r = 3$.

5-12 No cálculo do CRC, se a palavra de dados tem 5 *bits* e a palavra de código tem 8 *bits*, quantos 0s precisam ser adicionados à palavra de dados para gerar o dividendo? Qual é o tamanho do resto? Qual é o tamanho do divisor?

5-13 No cálculo do CRC, quais dos seguintes geradores (divisores) garantem a detecção de um erro de um único *bit*?

a. 101 **b.** 100 **c.** 1

5-14 No cálculo do CRC, quais dos seguintes geradores (divisores) garantem a detecção de um número ímpar de erros?

a. 10111 **b.** 101101 **c.** 111

5-15 No cálculo do CRC, escolhemos o gerador 1100101. Qual é a probabilidade de detectarmos um erro em rajada apresentando cada um dos seguintes comprimentos?

a. 5 **b.** 7 **c.** 10

5-16 Considere que estamos enviando conjuntos de dados de 16 *bits* de comprimento. Se dois conjuntos de dados forem trocados durante a transmissão, a soma de verificação (*checksum*) tradicional pode detectar esse erro? Explique.

5-17 O valor de uma soma de verificação (*checksum*) tradicional pode ser composto apenas por 0s (em binário)? Explique sua resposta.

5-18 Explique como o algoritmo de Fletcher (Figura 5.18) atribui pesos aos conjuntos de dados durante o cálculo da soma de verificação (*checksum*).

5-19 Explique por que há apenas um campo de endereço (e não dois) em um quadro HDLC.

5-20 Quais das seguintes opções são protocolos de acesso aleatório?

a. CSMA/CD

b. Varredura

c. TDMA

5-21 As estações em uma rede usando ALOHA puro enviam quadros de 1.000 *bits* de comprimento a uma taxa de transferência de 1 Mbps. Qual é o período vulnerável para essa rede?

5-22 Em uma rede usando ALOHA puro com G = 1/2, como a taxa de transferência é afetada em cada um dos seguintes casos?

a. G é aumentado para 1.

b. G é reduzido para 1/4.

5-23 Para entender a utilidade do parâmetro K na Figura 5.40, determine a probabilidade de que uma estação seja capaz de transmitir imediatamente em cada um dos seguintes casos:

a. Após uma falha.

b. Após três falhas.

5-24 Com base na Figura 5.30, como interpretamos a situação de *sucesso* em uma rede ALOHA?

5-25 Considere que o atraso de propagação em uma rede cujo meio é compartilhado seja de 5 μs e que o tempo de transmissão do quadro seja de 10 μs.

a. Quanto tempo leva para que o primeiro *bit* chegue ao destino?

b. Após a chegada do primeiro *bit*, quanto tempo leva para que o último *bit* chegue ao destino?

c. Quanto tempo a rede fica ocupada com esse quadro (vulnerável a colisões)?

5-26 Considere que o atraso de propagação em uma rede cujo meio é compartilhado seja de 3 μs e que o tempo de transmissão do quadro seja de 5 μs. A colisão pode ser detectada independentemente de onde ela ocorrer?

5-27 Duas estações em duas redes diferentes podem ter o mesmo endereço da camada de enlace?

5-28 Explique por que a colisão é um problema em protocolos de acesso aleatório, mas não o é em protocolos de acesso controlado ou de canalização.

5-29 Precisamos de um protocolo de acesso múltiplo quando usamos o serviço DSL fornecido pela companhia telefônica para acessar a Internet? Por quê?

5-30 O tamanho do pacote ARP é fixo? Explique.

5-31 Qual o tamanho de um pacote ARP quando o protocolo de rede é o IPv4 e o *hardware* é do tipo Ethernet?

5-32 Qual é o tamanho de um quadro Ethernet transportando um pacote ARP cujo tamanho é 28 *bytes*?

5-33 Em um quadro Ethernet, quais as diferenças entre o campo de preâmbulo e o campo SFD?

5-34 Para que serve uma placa de rede (NIC)?

5-35 Por que não é necessário usar CSMA/CD em uma LAN Ethernet *full-duplex*?

5-36 Compare as taxas de transferência de dados da Ethernet Padrão, Fast Ethernet, Ethernet Gigabit e Ethernet 10-Gigabit.

5-37 Quais são as implementações comuns da Ethernet Padrão?

5-38 Quais são as implementações comuns da Ethernet Gigabit?

5-39 O que é a tecnologia de modem discado? Liste alguns dos padrões comuns de modem discutidos neste capítulo e forneça suas taxas de transferência de dados.

5-40 Compare uma rede a cabo tradicional com uma rede híbrida fibra-coaxial, explicando suas diferenças e semelhanças.

5-41 Quatro estações são conectadas a um *hub* em uma rede Ethernet. As distâncias entre o *hub* e as estações são de 300 m, 400 m, 500 m e 700 m, respectivamente. Qual é o comprimento da rede que devemos considerar para o cálculo de T_p?

5-42 O que significa dizer que um *switch* de camada de enlace é capaz de filtrar o tráfego? Por que a filtragem é importante?

5-43 Como uma VLAN pode levar a uma economia de tempo e dinheiro?

5-44 Como uma VLAN pode reduzir o tráfego na rede?

5-45 Por que uma rede SONET é considerada uma rede síncrona?

5-46 Discuta as funções de cada camada do SONET.

5-47 No ATM, qual é a relação entre TPs, VPs e VCs?

Problemas

5-1 Faça o preenchimento de *byte* da seguinte carga útil de um quadro, considerando que E é o *byte* de escape, F é o marcador de fim de quadro e D é um *byte* de dados diferente de um caractere de escape ou de um marcador de fim de quadro.

D	E	D	D	F	D	D	E	E	D	F	D

5-2 Faça o preenchimento de *byte* da seguinte carga útil de um quadro:

0001111111001111101000111111111110000111

5-3 Qual é número máximo de *bits* corrompidos por um ruído em rajada de 2 ms quando os

dados são transmitidos a cada uma das seguintes taxas de transferência?

a. 1.500 bps
b. 12 kbps
c. 100 kbps
d. 100 Mbps

5-4 Considere que p seja a probabilidade de que um *bit* em uma unidade de dados seja corrompido durante a transmissão. Para uma unidade de dados de n *bits*, determine a probabilidade de que x *bits* sejam corrompidos em cada um dos casos a seguir.

a. $n = 8, x = 1, p = 0{,}2$
b. $n = 16, x = 3, p = 0{,}3$
c. $n = 32, x = 10, p = 0{,}4$

5-5 A operação de XOR (OU-Exclusivo) é uma das operações mais utilizadas no cálculo de palavras de código. Aplique a operação de XOR sobre as duas sequências de *bits* a seguir. Interprete os resultados.

a. $(10001) \oplus (10001)$
b. $(11100) \oplus (00000)$
c. $(10011) \oplus (11111)$

5-6 Na Tabela 5.1, o remetente envia a palavra de dados 10. Um erro em rajada de comprimento igual a 3 bits corrompe a palavra de código. O receptor é capaz de detectar o erro? Explique sua resposta.

5-7 Usando o código da Tabela 5.2, qual é a palavra de dados correspondente a cada uma das seguintes palavras de código recebidas?

a. 01011 **b.** 11111 **c.** 00000 **d.** 11011

5-8 Prove que o código representado pelas palavras de código a seguir não é linear. Para isso, basta que você encontre um caso que viole a linearidade.

{(00000), (01011), (10111), (11111)}

5-9 Qual é a distância de Hamming para cada um dos seguintes pares de palavras de código?

a. d (10000, 00000)
b. d (10101, 10000)
c. d (00000, 11111)
d. d (00000, 00000)

5-10 Embora seja possível provar formalmente que o código da Tabela 5.3 é tanto linear como cíclico, utilize os dois testes a seguir para provar parcialmente esse fato:

a. Teste a propriedade cíclica sobre a palavra de código 0101100.
b. Teste a propriedade linear sobre as palavras de código 0010110 e 1111111.

5-11 Com relação ao CRC-8 mostrado na Tabela 5.4, responda às seguintes perguntas:

a. Ele é capaz de detectar um erro de um único *bit*? Explique sua resposta.
b. Ele é capaz de detectar um erro em rajada de tamanho 6? Explique sua resposta.
c. Qual é a probabilidade de ele detectar um erro em rajada de tamanho 9?
d. Qual é a probabilidade de ele detectar um erro em rajada de tamanho 15?

5-12 Considerando que estamos usando paridade par, determine o *bit* de paridade para cada um dos seguintes conjuntos de dados.

a. 1001011
b. 0001100
c. 1000000
d. 1110111

5-13 Um simples *bit* de verificação de paridade, que é normalmente inserido no final da palavra (transformando um caractere ASCII de 7 *bits* em um *byte*), não é capaz de detectar um número par de erros. Por exemplo, erros em dois, quatro, seis ou oito *bits* não podem ser detectados usando essa técnica. Uma solução melhor é organizar os caracteres em uma tabela e criar paridades de linhas e colunas. Os *bits* de paridade das linhas são enviados juntamente com os *bytes*, enquanto os *bits* de paridade das colunas são enviados como um *byte* extra (Figura 5.88).

	C1	C2	C3	C4	C5	C6	C7	Paridades das linhas
R1	1	1	0	0	1	1	1	1
R2	1	0	1	1	1	0	1	1
R3	0	1	1	1	0	0	1	0
R4	0	1	0	1	0	0	1	1
Paridades das colunas	0	1	0	1	0	1	0	1

R*n*: Linha *n*
C*m*: Coluna *m*

Figura 5.88 Esquema para o Problema 5-13.

Mostre como os seguintes erros podem ser detectados:

a. Um erro na posição (L3, C3).
b. Dois erros nas posições (L3, C4) e (L3, C6).
c. Três erros nas posições (L2, C4), (L2, C5) e (R3, C4).

d. Quatro erros nas posições (L1, C2), (L1, C6), (L3, C2) e (L3, C6).

5-14 Dada a palavra de dados 101001111 e o divisor 10111, mostre como é feita a geração da palavra de código do CRC no lado do remetente (usando divisão binária).

5-15 Considere que um pacote é composto por apenas quatro palavras de 16 *bits*, $(A7A2)_{16}$, $(CABF)_{16}$, $(903A)_{16}$ e $(A123)_{16}$. Simule manualmente o algoritmo da Figura 5.17 para calcular a soma de verificação (*checksum*).

5-16 O cálculo tradicional da soma de verificação (*checksum*) precisa ser feito usando aritmética em complemento de um. Computadores e calculadoras atuais são projetados para realizar cálculos usando aritmética em complemento de dois. Uma forma de calcular a soma de verificação tradicional é somar os números usando aritmética em complemento de dois, determinar o quociente e o resto da divisão do resultado por 2^{16}, e então somar o quociente e o resto para obter a soma em complemento de um. A soma de verificação pode ser calculada subtraindo-se a soma do valor $2^{16} - 1$. Use esse método para calcular a soma de verificação dos quatro números a seguir: 43.689, 64.463, 45.112 e 59.683.

5-17 Esse problema mostra um caso especial na manipulação da soma de verificação. Um remetente tem dois conjuntos de dados para enviar: $(4567)_{16}$ e $(BA98)_{16}$. Qual é o valor da soma de verificação (*checksum*)?

5-18 Simule manualmente o algoritmo de Fletcher (Figura 5.18) para calcular a soma de verificação (*checksum*) dos *bytes* a seguir: $(2B)_{16}$, $(3F)_{16}$, $(6A)_{16}$ e $(AF)_{16}$. Mostre também que o resultado é uma soma de verificação ponderada.

5-19 Simule manualmente o algoritmo de Adler (Figura 5.19) para calcular a soma de verificação (*checksum*) das palavras a seguir: $(FBFF)_{16}$ e $(EFAA)_{16}$. Mostre também que o resultado é uma soma de verificação ponderada.

5-20 Como um exemplo prático, determine o valor do campo de soma de verificação (*checksum*) para o pequeno datagrama mostrado na Figura 5.89. Como a soma de verificação de um datagrama IP é calculado apenas para o cabeçalho, mostramos apenas os valores desses campos.

Figura 5.89 Esquema para o Problema 5-20.

5-21 Considere que um datagrama apresentando o cabeçalho da Figura 5.90 foi recebido. O valor da soma de verificação (*checksum*) é 65.207, conforme mostrado. Verifique se o cabeçalho foi corrompido durante a transmissão.

Cabeçalho IP

4	5	0	36
1		0	0
4	17		65207
180.124.168.110			
201.126.145.167			

Figura 5.90 Esquema para o Problema 5-21.

5-22 Um exemplo de uma soma de verificação (*checksum*) ponderada é o código ISBN-10 que podemos encontrar na capa traseira de alguns livros. No ISBN-10, há nove dígitos decimais que definem o país, a editora e o livro. O décimo dígito (mais à direita) é um dígito de verificação. O código $D_1D_2D_3D_4D_5D_6D_7D_8D_9C$, satisfaz a seguinte propriedade.

$$[(10 \times D_1) + (9 \times D_2) + (8 \times D_3) + \ldots + (2 \times D_9) + (1 \times C)] \bmod 11 = 0$$

Em outras palavras, os pesos são 10, 9, ..., 1. Se o valor calculado para C for 10, a letra X é usada em seu lugar. Ao substituir cada peso *w* pelo seu complemento usando aritmética módulo 11 ($11 - w$), pode-se demonstrar que o dígito verificador pode ser calculado conforme mostrado a seguir.

$$C = [(1 \times D_1) + (2 \times D_2) + (3 \times D_3) + \ldots + (9 \times D_9)] \bmod 11$$

Calcule o dígito verificador para o seguinte ISBN-10: **0-07-296775-C.**

5-23 Um código ISBN-13, uma nova versão do ISBN-10, é outro exemplo de uma soma de verificação (*checksum*) ponderada com 13 dígitos, no qual há 12 dígitos decimais que definem o livro e o último dígito é o dígito verificador. O código, $D_1D_2D_3D_4D_5D_6D_7D_8D_9D_{11}D_{12}C$ satisfaz a seguinte propriedade.

$$[(1 \times D_1) + (3 \times D_2) + (1 \times D_3) + \ldots + (3 \times D_{12}) + (1 \times C)] \bmod 10 = 0$$

Em outras palavras, os pesos são 1 e 3, alternadamente. Usando essa descrição, calcule o dígito verificador para o seguinte ISBN-13: **978-0-07-296775-C**.

5-24 Para criar uma fórmula de avaliação do desempenho de uma rede de acesso múltiplo, precisamos de um modelo matemático. Quando o número de estações em uma rede é muito grande, a distribuição de Poisson, dada por $p[x] = (e^{-\lambda} \times \lambda^x)/(x!)$, é usada. Nessa fórmula, $p[x]$ é a probabilidade de se gerar um número x de quadros em um período de tempo e λ é o número médio de quadros gerados durante o mesmo período. Usando a distribuição de Poisson:

a. Determine a probabilidade de que uma rede usando ALOHA puro gere x quadros durante o período vulnerável. Lembre-se que o período vulnerável para essa rede é duas vezes o intervalo de transmissão de quadros (T_{tr}).

b. Determine a probabilidade de que uma rede usando *slotted* ALOHA gere x quadros durante o período vulnerável. Lembre-se que o período vulnerável para essa rede é igual ao intervalo de transmissão de quadros (T_{tr}).

5-25 No problema anterior, foi utilizada a distribuição de Poisson para calcular a probabilidade de serem gerados x quadros, em um determinado período de tempo, em redes usando o ALOHA puro e o *slotted* ALOHA. Essa probabilidade foi calculada como $p[x] = (e^{-\lambda} \times \lambda^x)/(x!)$. Nesse problema, queremos calcular a probabilidade de que um quadro em tais redes chegue ao seu destino sem colidir com outros quadros. Para isso, é mais simples considerar que temos G estações, cada uma enviando em média um quadro durante o intervalo de transmissão de quadros (em vez de haver N quadros e uma média de G/N quadros sendo enviados durante o mesmo intervalo). Dessa forma, a probabilidade de sucesso para uma estação equivale à probabilidade de que nenhuma outra estação envie um quadro durante o período vulnerável.

a. Determine a probabilidade de que uma estação em uma rede usando ALOHA puro tenha sucesso em enviar um quadro durante um período vulnerável.

b. Determine a probabilidade de que uma estação em uma rede usando *slotted* ALOHA tenha sucesso em enviar um quadro durante um período vulnerável.

5-26 No problema anterior, descobrimos que a probabilidade de uma estação (em uma rede com G estações) enviar um quadro com sucesso durante um intervalo de tempo vulnerável é $P = e^{-2G}$ para o ALOHA puro e $P = e^{-2G}$ para o *slotted* ALOHA. Nesse problema, queremos determinar a vazão dessas redes, que corresponde à probabilidade de que alguma estação (dentre G estações) consiga enviar um quadro durante o período vulnerável.

a. Determine a vazão de uma rede usando ALOHA puro.

b. Determine a vazão de uma rede usando *slotted* ALOHA.

5-27 No problema anterior, mostramos que a vazão é $V = Ge^{-2G}$ para uma rede usando ALOHA puro e é $V = Ge^{-G}$ para o *slotted* ALOHA. Nesse problema, queremos determinar o valor de G em cada rede que leva à vazão máxima da rede, e então determinar o valor dessa vazão máxima. Isto pode ser feito se calcularmos o valor de G que faz com que a derivada de V em relação a G seja zero.

a. Determine o valor de G que leva à vazão máxima, e então encontre o valor da vazão máxima para uma rede usando ALOHA puro.

b. Determine o valor de G que leva à vazão máxima, e então encontre o valor da vazão máxima para uma rede usando *slotted* ALOHA.

5-28 Uma rede de acesso múltiplo com um grande número de estações pode ser analisada usando a distribuição de Poisson. Quando há um número reduzido de estações em uma rede, entretanto, precisamos utilizar outra abordagem para essa análise. Em uma rede com N estações, consideramos que a probabilidade de que cada estação tenha um quadro para enviar durante o intervalo de transmissão de quadros (T_{tr}) seja p. Nessa rede, uma estação envia seu quadro com sucesso se a estação tiver um quadro para enviar durante o período vulnerável e nenhuma outra estação tiver um quadro para enviar durante esse espaço de tempo.

a. Determine a probabilidade de que uma estação em uma rede usando ALOHA puro consiga enviar um quadro durante o período vulnerável.

b. Determine a probabilidade de que uma estação em uma rede usando *slotted* ALOHA consiga enviar um quadro durante o período vulnerável.

5-29 No problema anterior, determinamos a probabilidade de sucesso de uma estação que tenta enviar um quadro durante o período vulnerável. A vazão de uma rede com um número limitado de estações equivale à probabilidade de que qualquer estação (dentre N estações) seja capaz de enviar um quadro com sucesso. Em outras palavras, a vazão é a soma das N probabilidades de sucesso.

a. Determine a vazão de uma rede usando ALOHA puro.

b. Determine a vazão de uma rede usando *slotted* ALOHA.

5-30 No problema anterior, determinamos a vazão de uma rede usando ALOHA puro e *slotted* ALOHA como sendo $V = Np\,(1 - p)^{2(N-1)}$ e $V = Np\,(1 - p)^{(N-1)}$, respectivamente. Nesse problema, queremos determinar a vazão máxima com relação a p.

a. Determine o valor de p que maximiza a vazão de uma rede usando ALOHA puro, e calcule a vazão máxima quando N é um número muito grande.

b. Determine o valor de p que maximiza a vazão de uma rede usando *slotted* ALOHA, e calcule a vazão máxima quando N é um número muito grande.

5-31 Considere que existam apenas três estações ativas em uma rede usando *slotted* ALOHA: A, B e C. A probabilidade de que cada estação gere um quadro em uma faixa de tempo (*slot*) é dada respectivamente por $p_A = 0{,}2$; $p_B = 0{,}3$ e $p_C = 0{,}4$.

a. Qual é a vazão de cada estação?

b. Qual é a vazão da rede?

5-32 Considere que existam apenas três estações ativas em uma rede usando *slotted* ALOHA: A, B e C. A probabilidade de que cada estação gere um quadro em uma faixa de tempo (*slot*) é dada respectivamente por $p_A = 0{,}2$; $p_B = 0{,}3$ e $p_C = 0{,}4$.

a. Qual é a probabilidade de que alguma estação envie um quadro na primeira faixa de tempo?

b. Qual é a probabilidade de que a estação A envie um quadro com sucesso pela primeira vez na segunda faixa de tempo?

c. Qual é a probabilidade de que a estação C envie um quadro com sucesso pela primeira vez na terceira faixa de tempo?

5-33 Uma rede usando *slotted* ALOHA está operando na sua vazão máxima.

a. Qual é a probabilidade de que uma faixa de tempo esteja vazia?

b. Quantas faixas de tempo, em média, devem aparecer antes de observarmos uma faixa de tempo que esteja vazia?

5-34 Um parâmetro útil em uma LAN é o número de *bits* que cabem em um metro do meio de transmissão ($n_{b/m}$). Determine o valor de $n_{b/m}$, se a taxa de transferência de dados for de 100 Mbps e a velocidade de propagação no médio for de 2×10^8 m/s.

5-35 Outro parâmetro útil em uma LAN é o comprimento de *bit* do meio (L_b), que corresponde ao número de *bits* que o meio pode comportar em qualquer instante. Determine o comprimento de *bit* de uma LAN se a taxa de transferência de dados for de 100 Mbps e o comprimento médio em metros (L_m) para a comunicação entre duas estações for de 200 m. Considere que a velocidade de propagação no meio seja de 2×10^8 m/s.

5-36 Definimos o parâmetro a, o número de quadros que o meio entre duas estações pode comportar, como sendo $a = (T_p)/(T_{tr})$. Outra forma de definir esse parâmetro é $a = L_b/F_b$, onde L_b é o comprimento de *bit* do meio e F_b é o comprimento de quadro do meio. Mostre que essas duas definições são equivalentes.

5-37 Em uma rede em barramento usando CSMA e com uma taxa de transferência de dados de 10 Mbps, uma colisão ocorre 20 µs após o primeiro *bit* do quadro deixar a estação de origem. Qual deve ser o comprimento do quadro de modo que o remetente seja capaz de detectar a colisão?

5-38 Considere que existam apenas duas estações, A e B, em uma rede em barramento usando CSMA/CD. A distância entre as duas estações é de 2000 m e a velocidade de propagação é 2×10^8 m/s. Se a estação A começar a transmitir no instante de tempo t_1:

a. O protocolo permite que a estação B comece a transmitir no instante $t_1 + 8$ µs? Se a resposta for sim, o que vai acontecer?

b. O protocolo permite que a estação B comece a transmitir no instante $t_1 + 11$ µs? Se a resposta for sim, o que vai acontecer?

5-39 Considere que existam apenas duas estações, A e B, em uma rede em barramento usando CSMA/CD 1-persistente, na qual $T_p = 25{,}6$ µs e $T_{tr} = 51{,}2$ µs. A estação A tem um quadro que ela deseja enviar à estação B. O quadro é enviado duas vezes sem sucesso, mas é entregue com sucesso na terceira tentativa. Desenhe um diagrama com a linha do tempo que descreve esse problema. Considere que o valor de R seja 1 e 2, respectivamente, e ignore o tempo necessário para enviar um sinal de interferência (ver a Figura 5.40).

5-40 Para entender a razão pela qual precisamos ter um tamanho mínimo de quadro $T_{tr} = 2 \times T_p$ em uma rede usando CDMA/CD, suponha que temos uma rede em barramento com apenas duas estações, A e B, na qual $T_{tr} = 40$ µs e $T_p = 25$ µs. A estação A começa a enviar um quadro no instante $t = 0{,}0$ µs e a estação B começa a enviar um quadro no instante $t = 23$ µs. Responda às seguintes questões:

a. Os quadros colidem?

b. Se a resposta para o item a for sim, a estação A detecta a colisão?

c. Se a resposta para o item a for sim, a estação B detecta a colisão?

5-41 Em um barramento usando CSMA/CD 1-persistente no qual $T_p = 50$ µs e $T_{tr} = 120$ µs, existem duas estações, A e B. Elas começam a enviar quadros uma para a outra ao mesmo tempo. Como os quadros colidem, ambas as estações tentam retransmitir. A estação A sorteia R = 0, enquanto a estação B sorteia R = 1. Ignore qualquer outro atraso, incluindo o atraso relativo ao envio de sinais de interferência. Os quadros colidem novamente? Desenhe um diagrama com a linha de tempo que comprove a sua resposta. A geração de um número aleatório ajuda a evitar a colisão nesse caso?

5-42 A variável aleatória R (Figura 5.40) é projetada para gerar atrasos distintos em cada estação após a ocorrência de uma colisão. Para reduzir o número de colisões, espera-se que estações diferentes gerem valores distintos de R. Para isso, determine a probabilidade de que o valor de R seja o mesmo para duas estações após

a. A primeira colisão.

b. A segunda colisão.

5-43 Considere que tenhamos uma rede que usa *slotted* CSMA/CD. Cada estação nesta rede adota um período de contenção, durante o qual a estação abstém-se de acessar o canal compartilhado antes de tentar enviar um quadro. Considere que o período de contenção seja composto por faixas de tempo (*slots*) de contenção. No início de cada faixa, a estação verifica o meio. Se ele estiver livre, a estação envia seu quadro; se o canal estiver ocupado, a estação se abstém de enviar o quadro e espera até o início da próxima faixa. Em outras palavras, a estação espera, em média, por k faixas antes de enviar seu quadro, conforme mostra a Figura 5.91. Observe que o canal pode estar no estado de contenção, de transmissão ou no estado ocioso (quando nenhuma estação tiver um quadro para enviar). Entretanto, se N for um número muito grande, o estado ocioso acaba desaparecendo.

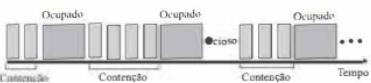



Figura 5.91 Esquema para o Problema 5-43.

a. Qual é a probabilidade de haver uma faixa livre (P_{livre}) se o número de estações for N e se cada estação tiver um quadro para enviar com probabilidade p?

b. Qual é o valor máximo dessa probabilidade quando N é um número muito grande?

c. Qual é a probabilidade de que a j-ésima faixa esteja livre?

d. Qual é o número médio de faixas, k, que uma estação deve esperar antes de encontrar uma faixa livre?

e. Qual é o valor de k quando N (o número de estações) é muito grande?

5-44 Embora o cálculo da vazão em uma rede CDMA/CD seja realmente complicado, podemos calcular a vazão máxima de uma rede *slotted* CDMA/CD usando a especificação descrita no problema anterior. Descobrimos que o número médio de faixas de contenção que uma estação precisa esperar é $k = e$ faixas. Com esse pressuposto, a vazão de uma rede usando *slotted* CDMA/CD é dada por

$V = (T_{tr})/$ (tempo durante o qual o canal fica ocupado com um quadro)

O tempo durante o qual o canal fica ocupado com um quadro corresponde ao tempo que precisamos esperar por uma faixa livre somado ao tempo para transmitir o quadro somado ao atraso de propagação até a estação perceber que não houve colisão. Considere que a duração de uma faixa de contenção seja $2 \times (T_p)$ e que $a = (T_p)/(T_{fr})$. Note que o parâmetro a é o número de quadros que ocupam o meio de transmissão. Determine a vazão de uma rede usando *slotted* CDMA/CD em função do parâmetro a.

5-45 Considere uma rede usando ALOHA puro com uma taxa de transferência de dados de 10 Mbps. Qual é o número máximo de quadros de 1.000 *bits* que podem ser enviados com sucesso por essa rede?

5-46 Em uma rede usando CDMA/CD com uma taxa de transferência de dados de 10 Mbps, o tamanho mínimo do quadro para a operação correta do processo de detecção de colisão foi calculado como sendo 512 *bits*.

Qual deve ser o tamanho mínimo do quadro se mantivermos o tamanho da rede constante, porém aumentarmos a taxa de transferência de dados para

a. 100 Mbps?

b. 1 Gbps?

c. 10 Gbps?

5-47 Qual é o valor hexadecimal equivalente ao endereço Ethernet a seguir?

```
01011010 00010001 01010101
00011000 10101010 00001111
```

5-48 Como o endereço Ethernet 1A:2B:3C:4D:5E:6F aparece na linha de transmissão, em binário?

5-49 Se um endereço Ethernet de destino é 07:01:02:03:04:05, qual é o seu tipo de endereço (*unicast*, *multicast* ou *broadcast*)?

5-50 Em uma rede usando CSMA/CD, há duas estações, A e B. Suponha que essas duas estações tenham um quadro para enviar durante o intervalo de colisão ($2 \times t_p$) e que elas o façam com probabilidade p_1 e p_2, respectivamente. Responda às seguintes perguntas, considerando $p_1 = 0,3$ e $p_2 = 0,4$:

a. Qual é a probabilidade de que estação A envie o quadro com sucesso?

b. Qual é a probabilidade de que estação B envie o quadro com sucesso?

c. Qual é a probabilidade de que um quadro seja enviado com sucesso?

5-51 Um roteador cujo endereço IPv4 é 125.45.23.12 e cujo endereço Ethernet é 23:45:AB:4F:67:CD recebeu um pacote destinado a uma estação cujo endereço IP é 125.11.78.10. Mostre os campos no pacote ARP de pedido enviado pelo roteador. Mostre também como fica o encapsulamento do pacote em um quadro Ethernet.

5-52 Na Figura 5.50, mostre as operações no lado de Bob quando a resposta é enviada de Bob para Alice.

5-53 A subcamada MAC da Ethernet recebe 42 *bytes* de dados da camada superior. Quantos *bytes* de enchimento (*padding*) devem ser adicionados aos dados?

5-54 Qual é a razão entre o tamanho dos dados úteis e o tamanho do quadro inteiro no menor quadro Ethernet?

5-55 Suponha que o comprimento de um cabo 10Base5 seja de 2500 m. Se a velocidade de propagação em um cabo coaxial grosso é de 200.000.000 m/s, quanto tempo leva para que um *bit* viaje de uma extremidade da rede até a outra? Considere um atraso de 10 μs no equipamento.

5-56 Que tipo de topologia é usado quando os clientes em uma região usam modems DSL para fins de transferência de dados? Explique.

5-57 Um *switch* da camada de enlace utiliza uma tabela de filtragem; já um roteador usa uma tabela de roteamento. Você pode explicar a diferença?

5.10 EXPERIMENTOS DE SIMULAÇÃO

5.10.1 *Applets*

Criamos alguns *applets* Java para demonstrar alguns dos principais conceitos discutidos neste capítulo. Recomendamos que o estudante ative esses *applets* no *site* www.mhhe.com/forouzan e examine cuidadosamente os protocolos em ação.

5.10.2 Experimentos de laboratório

Nesta seção, usamos o Wireshark para simular dois protocolos: o protocolo Ethernet e o protocolo ARP. As descrições completas desses experimentos de laboratório encontram-se no *site* www.grupoa.com.br.

Lab5-1 Neste laboratório, desejamos examinar o conteúdo de um quadro enviado pela camada de enlace de dados. Queremos descobrir o valor de diferentes campos, como os endereços MAC de origem e de destino, o valor do CRC, o valor do campo de protocolo (que mostra o tipo de carga útil sendo transportado pelo quadro), e assim por diante.

Lab5-2 Neste laboratório, desejamos examinar o conteúdo de um pacote ARP. Queremos capturar os pacotes ARP de pedido e de resposta. Os campos interessantes de examinarmos são aqueles que mostram que tipo de endereços de origem e de destino são usados em cada pacote.

5.11 TAREFAS DE PROGRAMAÇÃO

Para cada uma das tarefas a seguir, escreva um programa em uma linguagem de programação de sua preferência.

Prg5-1 Escreva e teste um programa que simule o preenchimento e a remoção de preenchimento de *byte*, conforme mostrado na Figura 5.5.

Prg5-2 Escreva e teste um programa que simule o preenchimento e a remoção de preenchimento de *bits*, conforme mostrado na Figura 5.7.

Prg5-3 Escreva e teste um programa que simule o diagrama de fluxo da Figura 5.17.

Prg5-4 Escreva e teste um programa que simule o diagrama de fluxo da Figura 5.18.

Prg5-5 Escreva e teste um programa que simule o diagrama de fluxo da Figura 5.19.

CAPÍTULO

6 REDES SEM FIO E IP MÓVEL

Discutimos redes com fios (também chamadas redes cabeadas ou redes guiadas) no Capítulo 5. Neste capítulo, apresentamos as redes sem fio. Discutiremos diversas tecnologias sem fio, incluindo LANs e outras redes sem fio, como telefonia celular, satélites e redes de acesso.

A natureza das redes sem fio é diferente daquela das LANs com fio, conforme veremos neste capítulo. A utilização de outras redes sem fio, como telefonia celular e comunicação via satélite, está aumentando na Internet. Dedicamos uma seção deste capítulo ao IP móvel: o uso do protocolo IP com estações móveis.

A tecnologia sem fio abrange, na realidade, tanto a camada de enlace de dados como a camada física da pilha de protocolos TCP/IP. Aqui, nos concentraremos principalmente nas questões relacionadas à camada de enlace de dados, adiando questões relativas à camada física até o Capítulo 7.

Este capítulo é dividido em três seções:

- A primeira seção apresenta as LANs sem fio e as compara com as LANs com fio, as quais foram discutidas no Capítulo 5. Discutimos, então, o projeto IEEE 802.11, padrão dominante em LANs sem fio. Em seguida, abordamos as LANs Bluetooth, que são usadas como LANs individuais em muitas aplicações. Finalmente, discutimos a tecnologia WiMAX, que é a correspondente sem fio às redes cabeadas de última milha como DSL ou redes a cabo.
- A segunda seção discute outras redes sem fio que podem ser classificadas como WANs sem fio ou redes de banda larga sem fio. Primeiramente, discutimos o método de acesso conhecido como canalização, utilizado em telefones celulares. Em seguida, abordamos a telefonia celular em si. Também tratamos de questões relativas aos satélites que fazem parte das tecnologias de conexão à Internet.
- A terceira seção cobre a área de IP móvel, que fornece acesso móvel à Internet. Começamos discutindo endereçamento, um grande problema no contexto de redes móveis. Em seguida, explicamos as três fases do acesso móvel. Finalmente, apontamos questões de ineficiência no móvel IP.

6.1 LANs SEM FIO

A comunicação sem fio é uma das tecnologias que vêm crescendo mais rapidamente na atualidade. A procura por dispositivos de conexão que não usam cabos está aumentando em todo lugar. LANs sem fio podem ser encontradas em universidades, edifícios comerciais e muitas áreas públicas.

6.1.1 Introdução

Antes de discutirmos um protocolo específico relacionado às LANs sem fio, explicaremos tais redes de forma geral.

Comparação arquitetural

Comecemos por uma comparação entre a arquitetura de redes locais com e sem fio para fornecermos uma ideia inicial do que precisamos observar no estudo das LANs sem fio.

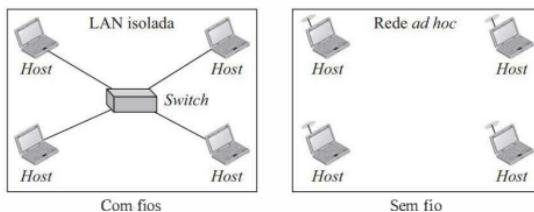
Meio físico

A primeira diferença que podemos observar entre uma LAN com fio e uma LAN sem fio refere-se ao meio de transmissão. Em uma LAN com fio, usamos fios (cabos) para conectar estações. No Capítulo 5, observamos a migração de tecnologias de acesso múltiplo para métodos de acesso ponto a ponto ao longo das gerações da Ethernet. Em uma LAN comutada, com um *switch* da camada de enlace, a comunicação entre os *hosts* é ponto a ponto e bidirecional (*full duplex*). Em uma LAN sem fio, o meio é o ar e o sinal é geralmente transmitido na forma de *broadcast*. Quando os *hosts* se comunicam entre si em uma LAN sem fio, eles estão compartilhando o mesmo meio físico (acesso múltiplo). Em uma situação muito rara, podemos criar uma comunicação ponto a ponto entre dois *hosts* sem fio usando uma largura de banda muito estreita e duas antenas direcionais. Nossa discussão neste capítulo, no entanto, concentra-se no cenário de acesso múltiplo ao meio, o que significa que precisamos usar protocolos de Controle de Acesso ao Meio (MAC – Medium Access Control).

Hosts

Em uma LAN com fio, um *host* está sempre conectado à sua rede por um ponto de acesso, tendo um endereço da camada de enlace fixo definido por sua placa de rede (NIC). Obviamente, um *host* pode se mover de um ponto da Internet para outro. Nesse caso, o seu endereço da camada de enlace permanece igual, porém o endereço na camada de rede mudará, conforme será discutido mais adiante, na seção sobre IP Móvel. Entretanto, antes que o *host* possa usar os serviços da Internet, ele precisa estar fisicamente conectado à Internet. Em uma LAN sem fio, um *host* não fica fisicamente conectado à rede; ele pode se mover livremente (conforme veremos mais adiante) e ainda assim usar os serviços prestados pela rede. Portanto, a mobilidade em uma rede com fio e em uma rede sem fio são questões totalmente diferentes, que tentamos esclarecer neste capítulo.

LANs isoladas

O conceito de uma LAN com fio isolada também difere do conceito de uma LAN sem fio isolada. Uma LAN com fio isolada consiste em um conjunto de estações conectadas entre si por meio de um *switch* da camada de enlace (na geração mais recente da Ethernet). Uma LAN sem fio isolada, denominada **rede *ad hoc*** * na terminologia de LAN sem fios, consiste em um conjunto de *hosts* que se comunicam livremente uns com os outros. O conceito de *switch* operando na camada de enlace não existe em LANs sem fio. A Figura 6.1 mostra duas LANs isoladas, uma com fios e outra sem.

Com fios

Sem fio

Figura 6.1 LANs isoladas: com fios *versus* sem fio.

* N. de T.: O termo *ad hoc* é uma expressão em latim que significa "para isto", sendo geralmente usada para designar uma solução concebida para um propósito específico, sem a intenção de que ela seja generalizada.

Conexão com outras redes

Uma LAN com fios pode ser conectada a outra rede ou a um conjunto de redes (por exemplo, a Internet) por meio de um roteador. Um LAN sem fio pode ser conectada a uma infraestrutura de rede com fios, a uma infraestrutura de rede sem fio ou a outra LAN sem fio. Nesta seção, discutimos a primeira situação: a conexão de uma LAN sem fio a uma infraestrutura de rede com fios. A Figura 6.2 mostra esses dois ambientes.

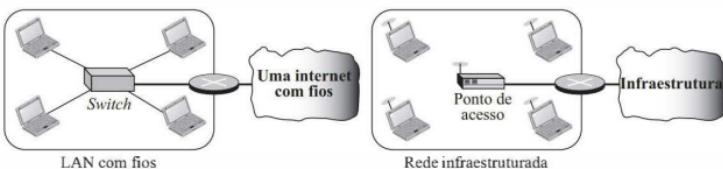



Figura 6.2 Conexão de uma LAN com fios e uma LAN sem fio a outras redes.

Nesse caso, a rede sem fio é denominada *rede infraestruturada* e a conexão à infraestrutura com fios, como a Internet, é feita por meio de um dispositivo conhecido como **Ponto de Acesso** (AP – Access Point). Perceba que o papel do AP é completamente distinto do papel de um *switch* da camada de enlace em um ambiente com fios. Um ponto de acesso interliga dois ambientes distintos entre si: um com fios e outro sem. A comunicação entre o AP e o *host* sem fio ocorre em um ambiente sem fio; a comunicação entre o AP e a infraestrutura ocorre em um ambiente com fios.

Migrando entre ambientes

A discussão anterior confirma o que aprendemos no capítulo anterior: tanto as LANs com fios como as LANs sem fio operam apenas nas duas camadas mais baixas da pilha de protocolos TCP/IP. Isto significa que se tivermos uma LAN com fios em um prédio que está conectado à Internet por meio de um roteador ou de um modem, tudo que é preciso fazer para migrarmos do ambiente com fios para o ambiente sem fio é substituir as placas de rede projetadas para um ambiente com fios por outras projetadas para um ambiente sem fio, além de substituir o *switch* da camada de enlace por um ponto de acesso. Com essa alteração, os endereços da camada de enlace mudarão (em razão da substituição dos NICs), porém os endereços da camada de rede (endereços IP) permanecerão os mesmos; estamos migrando de enlaces com fios para enlaces sem fio.

Características

Existem diversas características de LANs sem fio que não se aplicam a LANs com fios ou cujo efeito é desprezível e pode ser ignorado. Discutiremos algumas dessas características aqui com o objetivo de preparar o caminho para abordarmos protocolos de LANs sem fio.

Atenuação

A força dos sinais eletromagnéticos diminui rapidamente porque o sinal se dispersa em todas as direções; apenas uma pequena porção dessa energia atinge o receptor. A situação piora com emissores móveis que operam com baterias e, normalmente, apresentam uma fonte de energia limitada.

Interferência

Outro problema é que um receptor pode receber sinais não apenas do emissor desejado, mas também de outros emissores, caso eles estejam usando a mesma faixa de frequência.

Propagação por múltiplos caminhos

Um receptor pode receber mais que um sinal do mesmo remetente porque as ondas eletromagnéticas podem ser refletidas por obstáculos, como paredes, chão ou objetos. O resultado é que o receptor recebe sinais diferentes com fases distintas (porque eles viajaram por caminhos diferentes). Isto torna o sinal menos reconhecível pelo receptor.

Erros

Devido às características anteriores de uma rede sem fio, podemos esperar que os erros e a detecção de erros sejam problemas mais graves em uma rede sem fio do que em uma rede com fios. Se considerarmos o nível de erro como a medida da **relação Sinal/Ruído** (S/R, também conhecida como *signal-to-noise ratio*, ou SNR), podemos entender melhor o porquê da detecção de erros, da correção de erros e da retransmissão serem mais importantes em uma rede sem fio. Discutimos a S/R com mais detalhes no Capítulo 7, mas por ora é suficiente dizer que essa relação mede a proporção entre coisas boas e coisas ruins (sinal e ruído). Se a S/R for alta, significa que o sinal é mais forte do que o ruído (sinal indesejado), de modo que seremos capazes de converter o sinal nos dados de fato. Por outro lado, quando a S/R é baixa, significa que o sinal é corrompido pelo ruído e os dados não podem ser recuperados.

Controle de acesso

Provavelmente, a questão mais importante que precisamos discutir em uma LAN sem fio refere-se ao controle de acesso – como uma estação sem fio pode obter acesso ao meio compartilhado (o ar). Discutimos, no Capítulo 5, que a Ethernet padrão usava o algoritmo CSMA/CD. Nesse método, as estações disputam o acesso ao meio e enviam seus quadros se eles perceberem que o meio está ocioso. Se ocorrer uma colisão, ela é detectada e o quadro é enviado novamente. A detecção de colisão no CSMA/CD serve para dois propósitos. Se uma colisão for detectada, significa que o quadro não foi recebido e precisa ser reenviado. Se uma colisão não for detectada, pode-se considerar isto como uma confirmação de que o quadro foi recebido. O algoritmo CSMA/CD não funciona em LANs sem fio por três razões:

1. Para detectar uma colisão, uma estação precisa ser capaz de enviar e receber ao mesmo tempo (enviar o quadro e receber o sinal de colisão), o que significa que a estação precisa operar em um modo duplex. Muitas estações sem fio não têm potência suficiente para fazer isto (a energia é fornecida por baterias). Elas são capazes apenas de alternar entre enviar ou receber, mas não podem fazer ambos simultaneamente.
2. Devido ao problema da estação escondida, de acordo com o qual uma estação pode não estar ciente da transmissão de outra estação em razão de algum obstáculo ou a problemas de alcance, colisões podem ocorrer sem serem detectadas. A Figura 6.3 mostra um

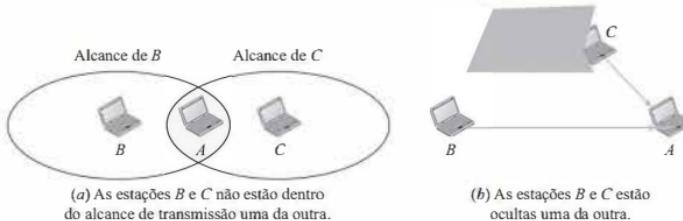



Figura 6.3 Problema da estação escondida.

exemplo do problema da estação escondida. O alcance de transmissão da estação *B* é representado pela forma ovalada à esquerda (esfera no espaço); todas as estações nessa região são capazes de ouvir qualquer sinal transmitido pela estação *B*. O alcance de transmissão da estação *C* é representado pela forma ovalada à direita (esfera no espaço); todas as estações localizadas nessa região são capazes de ouvir qualquer sinal transmitido por *C*. A estação *C* está fora do alcance de transmissão de *B*; de modo semelhante, a estação *B* está fora do alcance de transmissão de *C*. A estação *A*, no entanto, está na área coberta tanto por *B* como por *C*; ela é capaz de perceber qualquer sinal transmitido por *B* ou por *C*. A figura também mostra que o problema da estação escondida também pode ocorrer devido a um obstáculo.

Suponha que a estação *B* esteja enviando dados para a estação *A*. No meio dessa transmissão, a estação *C* também tem dados para enviar à estação *A*. No entanto, a estação *C* encontra-se fora do alcance de transmissão de *B*, de modo que as transmissões provenientes de *B* não chegam a *C*. Portanto, *C* acredita que o meio está ocioso. A estação *C* envia seus dados para a estação *A*, o que resulta em uma colisão em *A*, pois essa estação está recebendo dados tanto de *B* quanto de *C*. Nesse caso, podemos dizer que as estações *B* e *C* estão escondidas uma da outra em relação à estação *A*. Estações escondidas podem reduzir a capacidade de transmissão da rede devido à possibilidade de colisões.

3. A distância entre as estações pode ser grande. O desvanecimento do sinal poderia impedir que uma estação em uma extremidade ouvisse uma colisão na outra extremidade.

Para superar os três problemas anteriores, foi inventado o método Acesso Múltiplo com Verificação de Portadora/Prevenção de Colisão (CSMA/CA – Carrier Sense Multiple Access with Collision Avoidance) para LANs sem fio, que discutiremos mais adiante.

6.1.2 Projeto IEEE 802.11

O IEEE definiu as especificações para uma LAN sem fio, o padrão denominado IEEE 802.11, que abrange as camadas física e de enlace de dados. Em alguns países, incluindo os Estados Unidos e o Brasil, o público usa o termo *WiFi* (abreviatura de *Wireless Fidelity*, ou Fidelidade Sem Fio) como sinônimo de *LAN sem fio*. O termo WiFi, no entanto, refere-se a uma LAN sem fio que é certificada pela WiFi Alliance*, uma associação industrial global e sem fins lucrativos constituída por mais de 300 empresas dedicadas a promover o crescimento das LANs sem fio.

Arquitetura

O padrão 802.11 especifica dois tipos de serviços: Conjunto Básico de Serviços (BSS – Basic Service Set) e Conjunto Estendido de Serviços (ESS – Extended Service Set).

Conjunto Básico de Serviços

O IEEE 802.11 define o Conjunto Básico de Serviços (BSS – Basic Service Set) como a coleção de blocos construtivos de uma LAN sem fio. Um conjunto básico de serviços é composto por estações sem fio fixas ou móveis, bem como por uma estação-base central opcional conhecida como o Ponto de Acesso (AP – Access Point). A Figura 6.4 mostra dois cenários seguindo esse padrão.

* N. de T.: O principal objetivo da WiFi Alliance é certificar a conformidade aos padrões IEEE 802.11 para equipamentos de redes sem fio, de modo que eles operem adequadamente com os demais equipamentos já certificados.

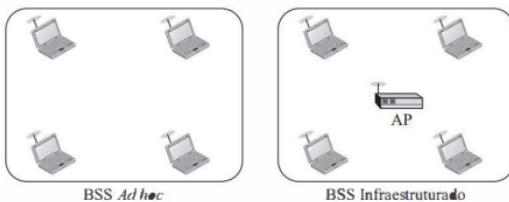



Figura 6.4 Conjunto Básico de Serviços (BSSs).

O BSS sem um AP é uma rede independente e incapaz de enviar dados para outros BSSs. Isto é conhecido como ***arquitetura ad hoc***. Nessa arquitetura, as estações podem formar uma rede sem a necessidade de um AP; podem localizar umas às outras e concordar em fazer parte de um BSS. O BSS com um AP é também conhecido como ***BSS infraestruturado***.

Conjunto Estendido de Serviços

Um **Conjunto Estendido de Serviços** (ESS – Extended Service Set) é composto por dois ou mais BSSs com APs. Nesse caso, os BSSs são conectados por meio de um Sistema de Distribuição (DS – Distribution System), que pode ser uma rede com fios ou sem. O sistema de distribuição interconecta os APs dos diferentes BSSs. O IEEE 802.11 não impõe restrições ao sistema de distribuição, que pode ser qualquer LAN IEEE, como uma LAN Ethernet. Perceba que o conjunto estendido de serviços usa dois tipos de estações: móveis e fixas. As móveis são estações normais dentro de um BSS. As fixas são estações do tipo AP que fazem parte de uma LAN com fios. A Figura 6.5 mostra um ESS.

Figura 6.5 Conjunto Estendido de Serviços (ESS).

Quando os BSSs estão conectados, as estações (dentro do alcance umas das outras) podem se comunicar sem utilizar um AP. No entanto, a comunicação entre uma estação em um BSS e uma estação fora daquele BSS ocorre por meio do AP. A ideia é semelhante à comunicação em uma rede celular (discutida mais adiante) se considerarmos cada BSS como uma célula e cada AP como uma estação-base. Perceba que uma estação móvel pode pertencer a mais de um BSS ao mesmo tempo.

Tipos de estação

O IEEE 802.11 define três tipos de estação com base na sua mobilidade em uma LAN sem fio: *sem transição*, *transição inter-BSS*, e *transição inter-ESS*. Uma estação com mobilidade do tipo sem transição pode ser fixa (não está se movendo) ou estar se movendo apenas dentro de um mesmo BSS. Uma estação com mobilidade do tipo transição inter-BSS pode se mover de um BSS para outro, porém sua movimentação está confinada ao interior de um mesmo ESS. Uma estação com mobilidade do tipo transição inter-ESS pode se mover de um ESS para outro. No entanto, o IEEE 802.11 não garante que a comunicação seja contínua durante a movimentação.

Subcamada MAC

O IEEE 802.11 define duas subcamadas MAC: Função de Coordenação Distribuída (DCF – Distributed Coordination Function) e Função de Coordenação Pontual (PCF – Point Coordination Function). A Figura 6.6 mostra a relação entre as duas subcamadas MAC, a subcamada LLC e a camada física. Discutimos as implementações da camada física mais adiante, mas, por ora, nos concentramos na subcamada MAC.

Figura 6.6 Camadas MAC no padrão IEEE 802.11

Função de Coordenação Distribuída

Um dos dois protocolos definidos pelo IEEE na subcamada MAC é denominado **Função de Coordenação Distribuída** (DCF – Distributed Coordination Function). O DCF usa o CSMA/CA como método de acesso.

CSMA/CA Como precisamos evitar colisões em redes sem fio dado que elas não podem ser detectadas, o Acesso Múltiplo com Verificação de Portadora/Prevenção de Colisão (CSMA/CA – Carrier Sense Multiple Access with Collision Avoidance) foi inventado para esse tipo de rede. Colisões são evitadas no CSMA/CA pela aplicação de três estratégias: espaço entre quadros, janela de contenção, e confirmações, conforme mostra a Figura 6.7. Discutimos os quadros RTS e CTS mais adiante.

- **Espaço entre quadros.** Primeiramente, as colisões são evitadas ao adiar a transmissão mesmo quando a estação detectar que o canal está ocioso. Quando a estação percebe que o canal está ocioso, ela não envia quadros imediatamente. Ela espera por um período de tempo denominado *Espaço entre Quadros*, ou *IFS* (Interframe Space). Embora o canal possa parecer ocioso quando verificado, uma estação distante pode já ter começado a transmitir. O sinal de tal estação distante ainda não chegou à estação verificando o meio. O tempo de IFS permite que a frente do sinal transmitido pela estação distante alcance a

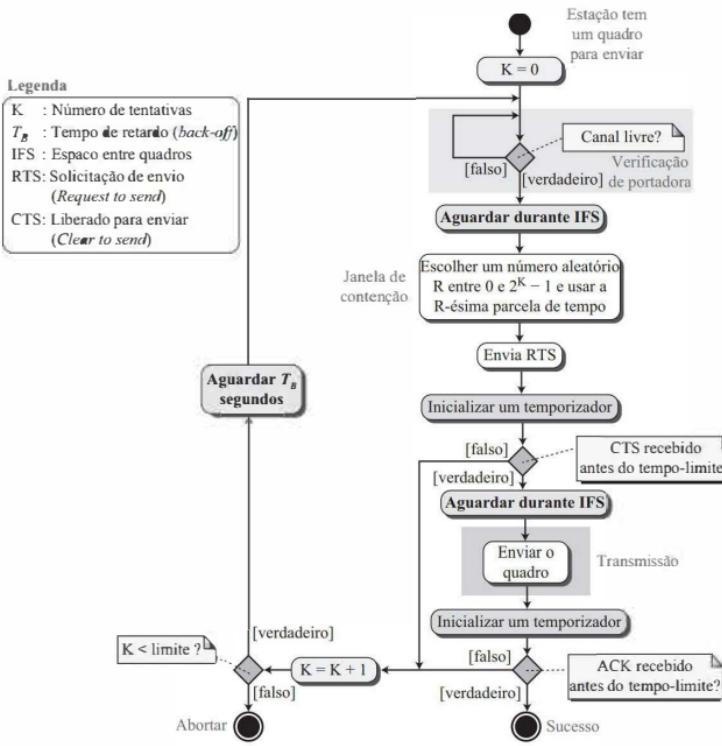



Figura 6.7 Diagrama de fluxo do CSMA/CA.

estação que está verificando o meio. Depois de esperar um intervalo igual a IFS, caso o canal continue ocioso, a estação pode enviar quadros, porém ela ainda precisa aguardar um intervalo de tempo igual ao tempo de contenção (descrito a seguir). A variável IFS também pode ser usada para priorizar estações ou tipos de quadros. Por exemplo, uma estação à qual é atribuído um IFS menor tem uma prioridade mais alta.

- **Janela de contenção.** A janela de contenção consiste em um período dividido em parcelas (*slots*). Uma estação que esteja pronta para enviar quadros escolhe um número aleatório de parcelas de tempo, que é usado como seu intervalo de espera. Esse número varia de acordo com a estratégia de retardo exponencial binário. Isso significa que seu valor é inicializado em um e então dobra cada vez que a estação não conseguir detectar a ociosidade do canal após o intervalo de IFS. É muito semelhante ao método *p*-persistente, exceto pelo fato de que um resultado aleatório define o número de parcelas de tempo utilizado pela estação que está aguardando. Um ponto interessante sobre a janela de contenção é que a estação precisa verificar o canal após cada parcela de tempo. No entanto, se encontrar o canal ocupado, ela não reinicia o processo;

apenas pausa o temporizador e o reinicia quando percebe que o canal está ocioso. Esse processo dá maior prioridade para estações que estejam esperando por mais tempo. Ver a Figura 6.8.

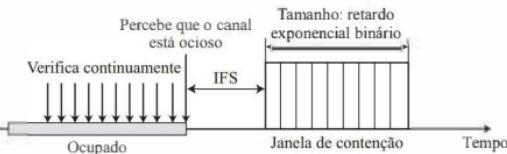



Figura 6.8 Janela de contenção.

- **Confirmação (ACK).** Mesmo tomando todas essas precauções, ainda pode ocorrer uma colisão que leve à destruição dos dados. Além disso, eles podem ser corrompidos durante a transmissão. A confirmação positiva e o temporizador com um tempo-limite podem ajudar a garantir que o receptor recebeu o quadro.

Linha de tempo da troca de quadros A Figura 6.9 mostra a troca de quadros de dados e de controle ao longo do tempo.

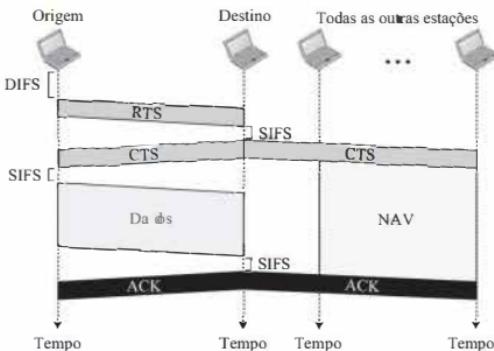



Figura 6.9 CSMA/CA e NAV.

1. Antes de enviar um quadro, a estação de origem verifica o meio, examinando o nível de energia na frequência portadora.

   a. A estação usa uma estratégia de persistência com retardo até que o canal esteja ocioso.

   b. Após a estação perceber que o meio encontra-se ocioso, ela espera por um período de tempo denominado Espaço entre Quadros DCF (DIFS – DCF Interframe Space*); em seguida, a estação envia um quadro de controle conhecido como *Solicitação de Envio* (RTS – Request To Send).

* N. de T.: A literatura também define o termo DIFS como Distributed Interframe Space, ou Espaço Distribuído entre Quadros.

2. Depois de receber o RTS e esperar um período de tempo conhecido como Espaço Curto entre Quadros (SIFS – Short Interframe Space) a estação de destino envia um quadro de controle denominado *Liberado para Enviar* (CTS – Clear to Send) para a estação de origem. Esse quadro de controle indica que a estação de destino está pronta para receber dados.
3. A estação de origem envia os dados depois de esperar um intervalo de tempo igual a SIFS.
4. A estação de destino, após esperar um intervalo de tempo igual a SIFS, envia uma mensagem de confirmação para mostrar que o quadro foi recebido. A confirmação é necessária nesse protocolo porque a estação não tem condições de verificar a chegada bem-sucedida dos seus dados ao destino. Em comparação, a ausência de colisão no CSMA/CD de certa forma indica ao emissor que os dados chegaram.

Vetor de alocação de rede Como as outras estações adiam o envio dos seus dados após uma estação adquirir o direito de acesso ao meio? Em outras palavras, como funciona o aspecto de *prevenção de colisão* desse protocolo? A chave para isso é um recurso denominado NAV.

Quando uma estação envia um quadro de RTS, este inclui o intervalo durante o qual aquela estação precisa ocupar o canal. As estações que são afetadas por essa transmissão criam um temporizador conhecido como **Vetor de Alocação de Rede** (NAV – Network Allocation Vector), que determina quanto tempo deve se passar até que essas estações tenham permissão para verificar por ociosidade no canal. Cada vez que uma estação acessa o sistema e envia um quadro de RTS, as outras estações iniciam seus NAVs. Em outras palavras, cada estação, antes de verificar o meio físico para determinar se ele está ocioso, primeiro verifica o seu NAV para determinar se ele expirou. A Figura 6.9 ilustra a ideia do NAV.

Colisão durante o estabelecimento da conexão O que acontece se houver uma colisão durante o intervalo de tempo no qual os quadros de controle RTS ou CTS estão em trânsito, intervalo comumente denominado período de estabelecimento de conexão ou período de *handshaking*? Duas ou mais estações podem tentar enviar quadros RTS ao mesmo tempo. Esses quadros de controle podem colidir. Entretanto, como não existe um mecanismo para detectar colisões, o emissor assume que houve uma colisão caso ele não receba um quadro CTS vindo do receptor. A estratégia de retardo é aplicada e o emissor tenta novamente.

Problema da estação escondida A solução para o problema da estação escondida é a utilização dos quadros de estabelecimento de conexão (RTS e CTS). A Figura 6.3 mostra também que a mensagem de RTS proveniente de *B* atinge *A*, porém não *C*. No entanto, como *B* e *C* estão ambos dentro da área de alcance de *A*, a mensagem de CTS, que especifica a duração da transmissão de dados de *B* para *A*, atinge *C*. A estação *C* sabe que alguma estação escondida está usando o canal e se abstém de transmitir durante o intervalo de tempo especificado na mensagem de CTS.

Função de Coordenação Pontual

A **Função de Coordenação Pontual** (PCF – Point Coordination Function) é um método de acesso opcional que pode ser implementado em uma rede infraestruturada (e não em uma rede *ad hoc*). Ela é implementada sobre o DCF e é usada principalmente em transmissões sensíveis a questões de tempo.

A PCF adota o método de acesso centralizado e sem contenção conhecido como varredura, discutido no Capítulo 5. O AP faz a varredura das estações que são capazes de ser verificadas, uma após a outra, enviando ao AP todos os dados que elas desejarem.

Para dar maior prioridade ao PCF com relação ao DCF, outro espaço entre quadros, denominado PIFS, foi definido. O Espaço Entre Quadros PCF (PIFS – PCF Interframe Space) tem um valor menor do que o DIFS. Isso significa que, caso uma estação queira utilizar apenas o DCF ao mesmo tempo em que um AP queira usar o PCF, o AP tem maior prioridade.

Devido à prioridade do PCF sobre o DCF, as estações que utilizam apenas o DCF podem não ser capazes de acessar o meio. Para evitar que isso ocorra, um *intervalo de repetição* foi

projetado para lidar tanto com o tráfego livre de contenção do PCF como com o tráfego baseado em contenção do DCF. O intervalo de repetição, que é repetido continuamente, começa com um quadro de controle especial denominado ***beacon frame*** ou **quadro de sinalização**. Quando as estações recebem o quadro de sinalização, elas inicializam seus respectivos NAV com a duração do período livre de contenção do intervalo de repetição. A Figura 6.10 mostra um exemplo de um intervalo de repetição.

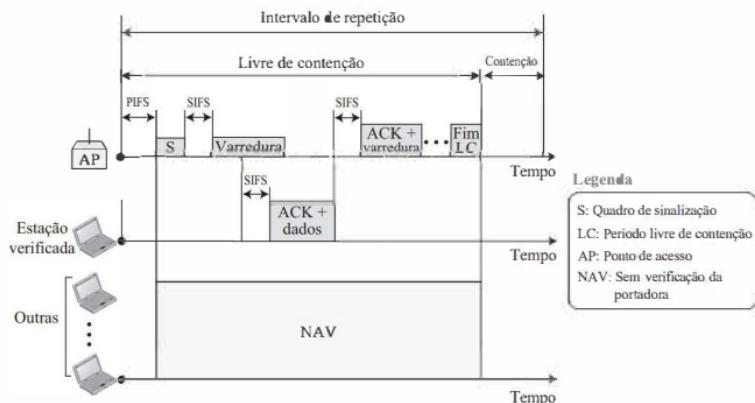



Figura 6.10 Exemplo de um intervalo de repetição.

Durante o intervalo de repetição, o CP (Controlador Pontual) pode enviar um quadro de verificação do processo de varredura, receber dados, enviar um ACK, receber um ACK ou qualquer combinação dessas ações (o 802.11 usa o mecanismo de carona, ou *piggybacking*). Ao final do período livre de contenção, o CP envia um quadro de "fim do LC" (fim do período livre de contenção) para permitir que as estações com acesso baseado em contenção usem o meio.

Fragmentação

O ambiente sem fio é muito ruidoso, razão pela qual os quadros são frequentemente corrompidos. Um quadro corrompido precisa ser retransmitido. O protocolo, portanto, recomenda a aplicação de fragmentação – a divisão de um quadro grande em quadros menores. É mais fácil reenviar um quadro pequeno do que um quadro grande.

Formato dos quadros

O quadro da subcamada MAC é composto por nove campos, conforme mostra a Figura 6.11.

- **Controle do Quadro (CQ).** O campo CQ tem 2 *bytes* de comprimento e especifica o tipo de quadro e algumas informações de controle. A Tabela 6.1 descreve os subcampos. Discutimos cada tipo de quadro mais adiante neste capítulo.
- **D.** Campo que especifica a duração da transmissão, usada para definir o valor do NAV. Em quadros de controle, ele define a identificação (ID) do quadro.

CQ	D	Endereço 1	Endereço 2	Endereço 3	CS	Endereço 4	Corpo do quadro	FCS
2 *bytes*	2 *bytes*	6 *bytes*	6 *bytes*	6 *bytes*	2 *bytes*	6 *bytes*	0 para 2.312 *bytes*	4 *bytes*

Versão do protocolo	Tipo	Subtipo	Para o DS	Do DS	Mais frags	Retrans	Ger. energia	Mais dados	WEP	Rsvd
2 *bits*	2 *bits*	4 *bits*	1 *bit*	1 *bit*	1 *bit*	1 *bit*	1 *bit*	1 *bit*	1 *bit*	1 *bit*

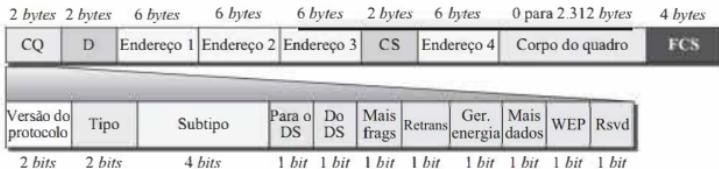



Figura 6.11 Formato dos quadros.

Tabela 6.1 Subcampos no campo CQ.

Campo	Explicação
Versão	A versão atual é 0
Tipo	Tipo da informação: gerenciamento (00), controle (01) ou dados (10)
Subtipo	Subtipo de cada tipo (ver Tabela 6.2)
Para o DS	Definido mais adiante
Do DS	Definido mais adiante
Mais frags	Quando seu valor é 1, significa que há mais fragmentos
Retrans	Quando seu valor é 1, significa que o quadro é uma retransmissão
Ger. Energia	Quando seu valor é 1, significa que a estação está em modo de gerenciamento de energia
Mais dados	Quando seu valor é 1, significa que a estação tem mais dados para enviar
WEP*	*Wired Equivalent Privacy*, ou Privacidade Equivalente à da Rede com Fios (cifração utilizada)
Rsvd	Reservado

- **Endereços.** Há quatro campos de endereços, cada um com 6 *bytes* de comprimento. O significado de cada campo de endereço depende do valor dos subcampos *Para o DS* e *Do DS*, conforme discutido mais adiante.
- **Controle de sequência.** Campo geralmente denominado campo CS, que contém um valor de 16 *bits*. Os primeiros quatro *bits* especificam o número do fragmento; os últimos 12 *bits* especificam o número de sequência, que é igual em todos os fragmentos.
- **Corpo do quadro.** Campo que pode ter entre 0 e 2.312 *bytes* e contém informações que dependem do tipo e do subtipo definido pelo campo de controle do quadro (CQ).
- **FCS.** O campo Sequência de Verificação de Quadros (FCS – Frame Check Sequence) tem 4 *bytes* de comprimento e contém uma sequência de detecção de erros do tipo CRC-32.

Tipos de quadros

Uma LAN sem fio especificada pelo IEEE 802.11 apresenta três categorias de quadros: quadros de gerenciamento, de controle e de dados.

* N. de T.: Em versões mais atuais do padrão 802.11, este subcampo foi renomeado para "Quadro Protegido" (Protected Frame).

Quadros de gerenciamento São usados na comunicação inicial entre as estações e os pontos de acesso.

Quadros de controle São usados para acessar o canal e confirmar a recepção de quadros. A Figura 6.12 mostra o formato desses quadros. Nos quadros de controle, o valor do campo de *tipo* é 01; os valores dos campos de *subtipo* para os quadros que discutimos aqui são mostrados na Tabela 6.2.

RTS:

2 *bytes*	2 *bytes*	6 *bytes*	6 *bytes*	4 *bytes*
CQ	D	Endereço 1	Endereço 2	FCS

CTS ou ACK:

2 *bytes*	2 *bytes*	6 *bytes*	4 *bytes*
CQ	D	Endereço 1	FCS

Figura 6.12 Quadros de controle.

Tabela 6.2 Valores dos subcampos nos quadros de controle.

Subtipo	Significado
1011	Solicitação de envio (RTS)
1100	Liberado para enviar (CTS)
1101	Confirmação (ACK)

Quadros de dados São usados para transportar dados e informações de controle.

Mecanismo de endereçamento

O mecanismo de endereçamento do IEEE 802.11 especifica quatro casos, definidos pelo valor de dois indicadores (*flags*) no campo CQ, ***Para o DS*** e ***Do DS***. Cada indicador pode assumir o valor 0 ou 1, resultando em quatro situações diferentes. A interpretação dos quatro endereços (*Endereço 1* a *Endereço 4*) no quadro de MAC depende do valor desses indicadores, conforme mostra a Tabela 6.3.

Tabela 6.3 Endereços.

Para o DS	Do DS	Endereço 1	Endereço 2	Endereço 3	Endereço 4
0	0	Destino	Origem	BSS-ID	N/A
0	1	Destino	AP emissor	Origem	N/A
1	0	AP receptor	Origem	Destino	N/A
1	1	AP receptor	AP emissor	Destino	Origem

Perceba que o Endereço 1 sempre corresponde ao endereço do próximo dispositivo que o quadro visitará. O Endereço 2 é sempre o endereço do dispositivo anterior, de onde o quadro veio. O Endereço 3 é o endereço da estação de destino final, caso isto não seja definido pelo Endereço 1, ou a estação emissora original, caso isto não seja definido pelo Endereço 2. O Endereço 4 corresponde ao endereço da estação emissora original quando o sistema de distribuição também é sem fio.

- **Caso 1: 00** Tem-se *Para o DS* = 0 e *Do DS* = 0. Isto significa que o quadro não está indo para um sistema de distribuição (*Para o DS* = 0) e não está vindo de um sistema de distribuição (*Do DS* = 0). Está indo de uma estação em um BSS para outra sem passar pelo sistema de distribuição. Os endereços são mostrados na Figura 6.13.
- **Caso 2: 01** Tem-se *Para o DS* = 0 e *Do DS* = 1. Isto significa que o quadro está vindo de um sistema de distribuição (*Do DS* = 1). Está vindo de um AP e indo para uma estação. Os endereços são mostrados na Figura 6.13. Observe que o Endereço 3 especifica o remetente original do quadro (em outro BSS).
- **Caso 3: 10** Tem-se *Para o DS* = 1 e *Do DS* = 0. Isto significa que o quadro está indo para um sistema de distribuição (*Para o DS* = 1). Está indo de uma estação para um AP. O ACK é enviado para a estação emissora original. Os endereços são mostrados na Figura 6.13. Perceba que o Endereço 3 contém o destino final do quadro no sistema de distribuição.
- **Caso 4: 11** Tem-se *Para o DS* = 1 e *Do DS* = 1. É o caso no qual o sistema de distribuição também é sem fio. O quadro está indo de um AP para outro em um sistema de distribuição sem fio. Aqui, precisamos de quatro endereços para especificar o emissor original, o destino final e os dois APs intermediários. A Figura 6.13 ilustra essa situação.

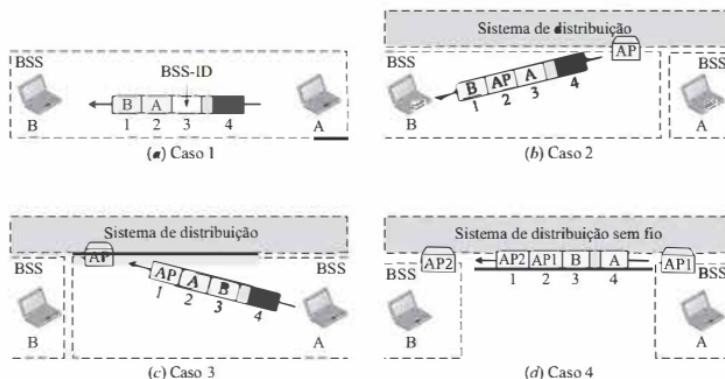



Figura 6.13 Mecanismos de endereçamento.

Problema da estação exposta

Discutimos anteriormente como resolver o problema da estação escondida. Um problema semelhante é conhecido como *problema da estação exposta*. Nele, uma estação se abstém de usar um canal quando ele está, na verdade, disponível. Na Figura 6.14, a estação *A* está transmitindo dados para a estação *B*. A estação *C* tem alguns dados para enviar à estação *D*, os quais podem ser enviados sem interferir com a transmissão de *A* para *B*. Entretanto, a estação *C* está exposta à transmissão proveniente de *A*; a estação *C* percebe que *A* está enviando dados e, portanto, se abstém de enviar seus próprios dados. Em outras palavras, *C* é muito conservadora e desperdiça a capacidade do canal. As mensagens de estabelecimento de conexão RTS e CTS não podem ajudar nesse caso. A estação *C* percebe o RTS enviado por *A* e se abstém de enviar, embora a comunicação entre *C* e *D* não possa causar uma colisão na região entre *A* e *C*; a estação *C* não tem como saber que a transmissão da estação *A* não afeta a região entre *C* e *D*.

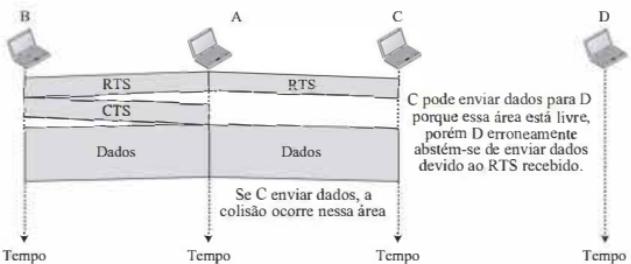



Figura 6.14 Problema da estação exposta.

Camada física

Discutimos seis especificações, conforme mostra a Tabela 6.4. Todas elas, exceto pelo infravermelho, operam na banda *Industrial, Científica e Médica* (ISM – Industrial, Scientific and Medical), que especifica três bandas não licenciadas nas seguintes faixas: 902-928 MHz, 2.400-4.835 GHz e 5.725-5.850 GHz.

Tabela 6.4 Especificações

IEEE	Técnica	Banda	Modulação	Taxa de Transmissão (Mbps)
802.11	FHSS	2.400-4.835 GHz	FSK	1 e 2
	DSSS	2.400-4.835 GHz	PSK	1 e 2
	Nenhuma	Infravermelho	PPM	1 e 2
802.11a	OFDM	5.725-5.850 GHz	PSK ou QAM	6 a 54
802.11b	DSSS	2.400-4.835 GHz	PSK	5,5 e 11
802.11g	OFDM	2.400-4.835 GHz	Diferente	22 e 54
802.11n	OFDM	5.725-5.850 GHz	Diferente	600

IEEE 802.11 FHSS

O IEEE 802.11 FHSS utiliza o método Espalhamento Espectral por Saltos em Frequências (FHSS – Frequency Hopping Spread Spectrum), discutido no Capítulo 7. O FHSS usa a banda ISM de 2.400-4.835 GHz. A banda é dividida em 79 sub-bandas de 1 MHz (e algumas bandas de guarda). Um gerador de números pseudoaleatórios seleciona a sequência de saltos. A técnica de modulação nessa especificação pode ser a Modulação por Chaveamento de Frequência (FSK – Frequency Shift Keying) de dois níveis ou a FSK de quatro níveis com 1 ou 2 *bits/baud*, o que resulta em uma taxa de transferência de dados de 1 ou 2 Mbps, conforme mostra a Figura 6.15.

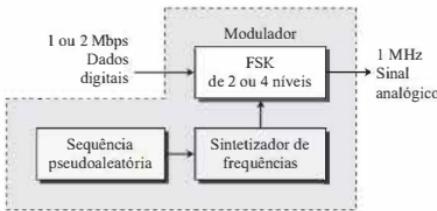



Figura 6.15 Camada física do IEEE 802.11 FHSS.

IEEE 802.11 DSSS

O IEEE 802.11 DSSS utiliza o método Espalhamento Espectral por Sequência Direta (DSSS – Direct Sequence Spread Spectrum), discutido no Capítulo 7. O DSSS usa a banda ISM de 2.400-4.835 GHz. A técnica de modulação nessa especificação é a Modulação por Chaveamento de Fase (PSK – Phase Shift Keying) a 1 Mbaud/s. O sistema permite 1 ou 2 *bits/baud*, usando BPSK (PSK Binário) ou QPSK (PSK em Quadratura), o que resulta em uma taxa de transferência de dados de 1 ou 2 Mbps, conforme mostra a Figura 6.16.

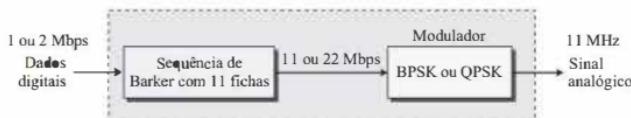



Figura 6.16 Camada física do IEEE 802.11 DSSS.

IEEE 802.11 Infravermelho

O IEEE 802.11 Infravermelho utiliza luz infravermelha na faixa de 800 a 950 nm. A técnica de modulação é denominada **Modulação por Posição de Pulso** (PPM – Pulse Position Modulation). Para uma taxa de transferência de dados de 1 Mbps, uma sequência de 4 *bits* é primeiramente mapeada em uma sequência de 16 *bits* na qual apenas um *bit* vale 1 e os *bits* restantes são fixados em 0. Para uma taxa de transferência de dados de 2 Mbps, uma sequência de 2 *bits* é primeiramente mapeada em uma sequência de 4 *bits* na qual apenas um *bit* vale 1 e os *bits* restantes são fixados em 0. As sequências mapeadas são, então, convertidas em sinais ópticos; a presença de luz significa um *bit* 1, enquanto a ausência de luz significa 0. Ver Figura 6.17.

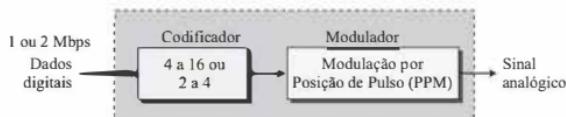



Figura 6.17 Camada física do IEEE 802.11 infravermelho.

IEEE 802.11a OFDM

O IEEE 802.11a OFDM descreve o método **Multiplexação por Divisão de Frequências Ortogonais** (OFDM – Orthogonal Frequency-Division Multiplexing) para a geração de sinais em uma banda ISM de 5.725-5.850 GHz. O OFDM é semelhante ao FDM, discutido no Capítulo 7, com uma diferença principal: todas as sub-bandas são utilizadas por um emissor em um dado instante. Os emissores disputam uns com os outros pelo acesso à camada de enlace de dados. A banda é dividida em 52 sub-bandas, com 48 sub-bandas para enviar 48 grupos de *bits* de cada vez e quatro sub-bandas para informações de controle. A divisão da banda em sub-bandas reduz os efeitos de interferências. Se as sub-bandas forem utilizadas de forma aleatória, pode-se também obter maior segurança. O OFDM usa a técnica de PSK ou de Modulação por Amplitude em Quadratura (QAM – Quadrature Amplitude Modulation) para a modulação. As taxas de transferência de dados mais comuns são 18 Mbps (PSK) e 54 Mbps (QAM).

IEEE 802.11b DSSS

O IEEE 802.11b DSSS descreve o método **Espalhamento Espectral por Sequência Direta de Alta Taxa** (HR-DSSS – High-Rate Direct-Sequence Spread Spectrum) para a geração de sinais na banda ISM de 2.400-4.835 GHz. O HR-DSSS é semelhante ao DSSS, exceto pelo método de codificação, conhecido como Chaveamento de Código Complementar (CCK – Complementary Code Keying). O CCK codifica 4 ou 8 *bits* em um símbolo CCK. Para ser retrocompatível com o DSSS, o HR-DSSS especifica quatro taxas de transferência de dados: 1 Mbps, 2 Mbps, 5,5 Mbps e 11 Mbps. As duas primeiras utilizam as mesmas técnicas de modulação que o DSSS. A versão de 5,5 Mbps usa BPSK e transmite dados a 1,375 Mbaud/s com codificação CCK de 4 *bits*. A versão de 11 Mbps usa o QPSK e transmite dados a 1,375 Mbps com codificação CCK de 8 *bits*. A Figura 6.18 ilustra a técnica de modulação desse padrão.

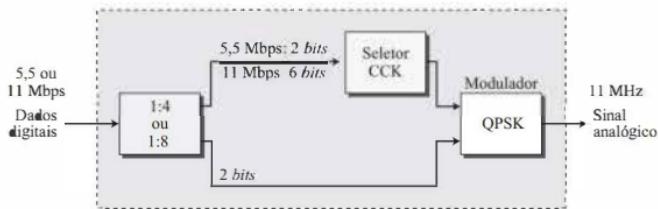



Figura 6.18 Camada física do IEEE 802.11b.

IEEE 802.11g

Essa nova especificação aplica técnicas de Correção Antecipada de Erros (FEC – Forward Error Correction) e OFDM, usando a banda ISM de 2.400-4.835 GHz. A técnica de modulação atinge uma taxa de transferência de dados de 22 ou 54 Mbps. Ela possui compatibilidade com a 802.11b, mas a técnica de modulação usada é o OFDM.

IEEE 802.11n

Uma evolução do projeto 802.11 é o chamado 802.11n (a nova geração de LANs sem fio). O objetivo dessa especificação é aumentar a vazão de LANs sem fio 802.11. O novo padrão tem como foco não apenas a maior taxa de *bits*, mas também elimina alguns elementos desnecessários. O padrão utiliza uma técnica conhecida como **Múltiplas Entradas e Múltiplas Saídas** (MIMO –

Multiple-Input Multiple-Output)[*] para superar o problema de ruído em LANs sem fio. A ideia é que se pudermos enviar múltiplos sinais de saída e receber múltiplos sinais de entrada, conseguimos eliminar o ruído mais facilmente. Algumas implementações desse projeto podem atingir uma taxa de transferência de dados de até 600 Mbps.

6.1.3 Bluetooth

O **Bluetooth** é uma tecnologia de LAN sem fio projetada para conectar dispositivos de diferentes funções, como telefones, *notebooks*, computadores (*desktops* e *laptops*), câmeras, impressoras e até mesmo máquinas de café, quando eles estiverem a uma pequena distância uns dos outros. Uma LAN Bluetooth é uma rede *ad hoc*, o que significa que a rede é formada espontaneamente; os dispositivos, também conhecidos como *gadgets*, localizam uns aos outros e formam uma rede denominada *piconet*. Uma LAN Bluetooth pode também ser conectada à Internet se um dos dispositivos tiver capacidade para tal. Uma LAN Bluetooth, por natureza, não pode ser grande. Se houver muitos dispositivos tentando se conectar, o resultado é caótico.

A tecnologia Bluetooth tem diversas aplicações. Dispositivos periféricos, como um mouse ou teclado sem fio, podem se comunicar com o computador usando essa tecnologia. Dispositivos de monitoramento podem se comunicar com dispositivos sensores em um pequeno centro médico. Os de segurança doméstica podem usar essa tecnologia para conectar diferentes sensores ao controlador central de segurança. Os participantes de uma conferência podem sincronizar seus computadores portáteis durante a conferência.

O Bluetooth foi originalmente criado como um projeto da companhia Ericsson. Ele foi nomeado em homenagem a Harald Blaatand, o rei da Dinamarca (940-981) que unificou a Dinamarca e a Noruega. *Blaatand* se traduz como Bluetooth[**] em inglês.

Atualmente, a tecnologia Bluetooth corresponde à implementação de um protocolo definido pelo padrão IEEE 802.15. Esse padrão define uma Rede Pessoal (PAN – Personal Area Network) sem fio, operável em uma área do tamanho de um quarto ou de uma sala.

Arquitetura

O Bluetooth especifica dois tipos de redes: *piconet* e *scatternet*.

Piconets

Uma rede Bluetooth é denominada uma ***piconet***, termo que pode ser entendido como "rede minúscula". Uma *piconet* pode ter até oito estações, uma das quais é conhecida como *primária*; as estações restantes são *secundárias*. Todas as estações secundárias sincronizam seus relógios e sua sequência de saltos com a estação primária. Perceba que uma *piconet* pode ter apenas uma estação primária. A comunicação entre as estações primária e secundárias pode ser um para um ou um para muitos. A Figura 6.19 mostra uma *piconet*.

Embora uma *piconet* possa ter um máximo de sete estações secundárias, estações secundárias adicionais podem estar no *estado inativo*, também conhecido como *estado estacionado*. Uma estação secundária no estado inativo está sincronizada com a estação primária, porém ela não pode participar da comunicação até que saia do estado inativo e vá para o estado ativo. Como apenas oito estações podem estar ativas em uma *piconet*, ativar uma estação que esteja no estado inativo pode fazer com que uma estação ativa vá para o estado inativo.

[*] N. de T.: O MIMO normalmente é obtido empregando-se múltiplas antenas em cada um dos equipamentos.

[**] N. de T.: Literalmente, a palavra Bluetooth pode ser traduzida como "dente azul" em português.

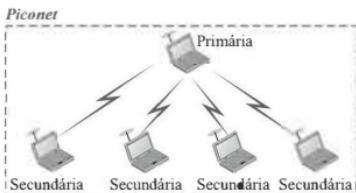



Figura 6.19 *Piconet*

Scatternet

As *piconets* podem ser combinadas para formar um tipo de rede denominado ***scatternet***, termo que pode ser entendido como "rede espalhada". Uma estação secundária em uma *piconet* pode ser a estação primária de outra *piconet*. Essa estação pode receber mensagens provenientes da estação primária da primeira *piconet* (atuando como uma estação secundária) e, atuando como uma estação primária, entregá-las às estações secundárias na segunda *piconet*. Uma estação pode participar de duas *piconets*. A Figura 6.20 ilustra uma *scatternet*.

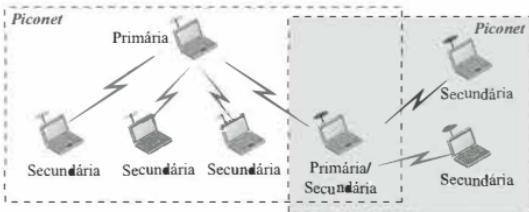



Figura 6.20 *Scatternet*

Dispositivos Bluetooth

Um dispositivo Bluetooth tem embutido um transmissor de rádio de curto alcance. A taxa de transferência de dados da versão 1 do protocolo* é de 1 Mbps, com uma largura de banda de 2.4 GHz. Isto significa que é possível haver interferências entre LANs sem fio usando o padrão IEEE 802.11b e LANs Bluetooth.

Camadas Bluetooth

O Bluetooth usa diversas camadas que não coincidem exatamente com as do modelo de Internet que definimos neste livro. A Figura 6.21 mostra essas camadas.

L2CAP

O **Protocolo de Adaptação e Controle de Enlace Lógico** (L2CAP – Logical Link Control and Adaptation Protocol, onde L2 significa LL), é aproximadamente equivalente à subcamada LLC em

* N. de T.: O Bluetooth encontra-se atualmente na versão 4.0 e pode atingir taxas de transferência de até 24Mbps.

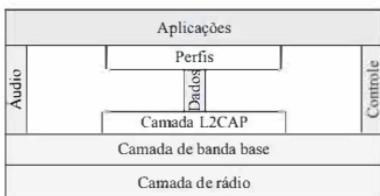



Figura 6.21 Camadas Bluetooth.

Figura 6.22 Formato do pacote de dados L2CAP.

redes locais. Ele é usado para a troca de dados em um canal ACL (*Asynchronous Connectionless Link*, ou Enlace Assíncrono Sem Conexão); canais SCO (*Synchronous Connection-Oriented, ou* Síncrono Orientado à Conexão) não usam o L2CAP. A Figura 6.22 mostra o formato do pacote de dados nesse nível.

O campo de comprimento, de 16 *bits*, especifica o tamanho dos dados, em *bytes*, proveniente das camadas superiores. Os dados podem ser compostos por até 65.535 *bytes*. O identificador do canal (CID – channel ID) é um identificador único para o canal virtual criado nesse nível (ver a seguir).

O L2CAP desempenha tarefas específicas: multiplexação, segmentação e remontagem, qualidade de serviço (QoS) e gerenciamento de grupo.

Multiplexação O L2CAP pode fazer multiplexação. No lado do emissor, o L2CAP recebe os dados provenientes de um dos protocolos de camada superior, os enquadra e entrega o resultado à camada de banda base. No lado do receptor, o L2CAP recebe um quadro da camada de banda base, extrai os dados e os entrega ao protocolo apropriado da camada superior. Isto cria uma espécie de canal virtual que discutiremos em capítulos posteriores para protocolos de mais alto nível.

Segmentação e remontagem O tamanho máximo do campo de carga útil na camada de banda base é 2.774 *bits*, ou 343 *bytes*. Isto inclui 4 *bytes* que delimitam o pacote e definem o tamanho do pacote. Logo, o tamanho do pacote proveniente de uma camada superior pode ser no máximo 339 *bytes*. No entanto, as camadas de aplicação algumas vezes precisam enviar um pacote de dados que pode ter até 65.535 *bytes* (um pacote vindo da Internet, por exemplo). O L2CAP divide esses pacotes grandes em segmentos e adiciona informações extras para especificar a localização dos segmentos no pacote original. O L2CAP segmenta o pacote na origem e o remonta no destino.

QoS O Bluetooth permite que as estações especifiquem um nível de qualidade de serviço. Discutiremos qualidade de serviço no Capítulo 8. Por ora, é suficiente saber que, se um nível de qualidade de serviço não for definido, o Bluetooth emprega por padrão o chamado *serviço de melhor esforço*; ele dará o melhor de si de acordo com as circunstâncias.

Gerenciamento de grupo Outra funcionalidade do L2CAP é permitir que os dispositivos criem um tipo de endereçamento lógico entre eles. Isto é similar à transmissão *multicast*. Por exemplo, dois ou três dispositivos secundários podem fazer parte de um grupo *multicast* para receber dados provenientes do dispositivo primário.

Camada de banda base

A camada de banda base é aproximadamente equivalente à subcamada MAC em redes locais. O método de acesso é o TDMA (discutido mais tarde). As estações primária e secundárias comunicam-se umas com as outras usando parcelas discretas (*slots*) de tempo. O comprimento de uma parcela de tempo é exatamente igual ao tempo de permanência em uma frequência antes de um salto, que é de 625 μs. Isto significa que, durante o tempo em que uma frequência é usada, uma estação primária envia um quadro a uma secundária ou uma estação secundária envia um quadro a uma primária. Perceba que a comunicação se dá apenas entre uma estação primária e uma estação secundária; estações secundárias não podem se comunicar diretamente.

TDMA O Bluetooth utiliza uma forma de TDMA denominada **TDMA com Duplexação por Divisão de Tempo** (TDD-TDMA – Time-Division Duplex TDMA). O TDD-TDMA é uma espécie de comunicação *half-duplex* na qual o emissor e o receptor enviam e recebem dados, porém não ao mesmo tempo (*half-duplex*); no entanto, a comunicação em cada direção utiliza sequências de salto diferentes. Isto é semelhante ao que acontece com *walkie-talkies*, que usam diferentes frequências portadoras.

- **Comunicação com uma única estação secundária.** Se a *piconet* tiver apenas uma estação secundária, a operação do TDMA é muito simples. O tempo é dividido em parcelas discretas de 625 μs. A estação primária usa as parcelas de numeração par (0, 2, 4, ...); já a estação secundária usa as parcelas ímpares (1, 3, 5, ...). O TDD-TDMA permite que as estações primária e secundária se comuniquem no modo *half-duplex*. Na parcela 0, a estação primária envia dados e a secundária os recebe; na parcela 1, a estação secundária envia dados e a primária os recebe. O ciclo é então repetido. A Figura 6.23 ilustra esse conceito.

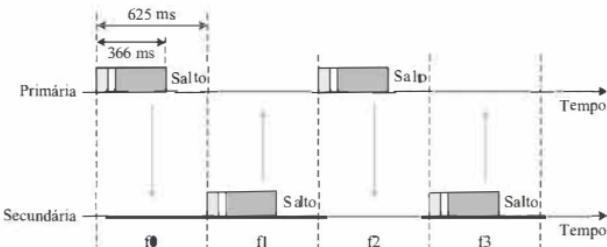



Figura 6.23 Comunicação com uma única estação secundária.

- **Comunicação com múltiplas estações secundárias.** O processo é um pouco mais complicado se houver mais do que uma estação secundária na *piconet*. Novamente, a estação primária usa as parcelas de tempo pares, porém uma estação secundária envia dados na próxima parcela de tempo ímpar se o pacote na parcela de tempo anterior foi endereçado àquela estação. Todas as estações secundárias verificam o meio em parcelas de tempo pares, mas apenas uma delas envia dados na parcela de tempo ímpar seguinte. A Figura 6.24 ilustra tal cenário.

Discutiremos essa figura em mais detalhes.

1. Na parcela de tempo 0, a estação primária envia um quadro para a estação secundária 1.

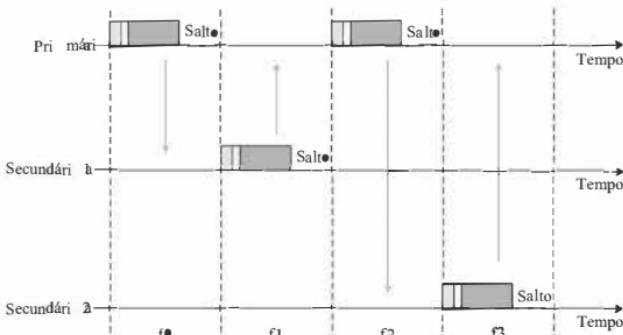



Figura 6.24 Comunicação com múltiplas estações secundárias.

2. Na parcela de tempo 1, apenas a estação secundária 1 envia uma quadro para a estação primária porque o quadro anterior foi endereçado à estação secundária 1; as outras estações secundárias ficam em silêncio.
3. Na parcela de tempo 2, a estação primária envia um quadro para a estação secundária 2.
4. Na parcela de tempo 3, apenas a estação secundária 2 envia um quadro para a estação primária porque o quadro anterior foi endereçado à estação secundária 2; as outras estações secundárias ficam em silêncio.
5. O ciclo continua.

Podemos dizer que esse método de acesso é semelhante a uma operação de varredura/seleção com reservas. Quando a estação primária seleciona uma estação secundária, ela também verifica se esta tem algo para enviar. A próxima parcela de tempo é reservada para que aquela estação envie seu quadro. Se a estação secundária não tiver quadros para enviar, o canal permanece em silêncio.

Canais Existem dois tipos de canais que podem ser criados entre uma estação primária e uma secundária: canais SCO e ACL.

- Um canal SCO (*Synchronous Connection-Oriented,* ou **Síncrono Orientado à Conexão**) é usado quando evitar a latência (atraso na entrega dos dados) é mais importante do que preservar a integridade dos dados (realizar a entrega livre de erros). Em um canal SCO, uma ligação física é criada entre as estações primária e secundária por meio da reserva de parcelas de tempo específicas em intervalos regulares. A unidade básica da conexão são duas parcelas, uma em cada direção. Se um pacote for corrompido, ele nunca é retransmitido. O canal SCO é usado para áudio em tempo real, cenário no qual evitar atrasos é muito importante. Uma estação secundária pode criar até três canais SCO com a estação primária, enviando áudio digitalizado (PCM) a 64 kbps em cada canal.
- Um canal ACL (*Asynchronous Connectionless Link,* ou **Enlace Assíncrono Sem Conexão**) é usado quando a integridade dos dados é mais importante do que evitar latência. Nesse tipo de canal, se uma carga útil encapsulada no quadro for corrompida, ela é retransmitida. Uma estação secundária retorna um quadro ACL na parcela de tempo ímpar disponível se a parcela de tempo anterior tiver sido endereçada àquela estação. O canal ACL pode usar um, três, ou mais parcelas de tempo, podendo alcançar uma taxa de transferência de dados máxima de 721 kbps.

Formato dos quadros Um quadro na camada de banda base pode ser de um dos seguintes três tipos: uma parcela, três parcelas ou cinco parcelas. Uma parcela de tempo, conforme discutido anteriormente, corresponde a 625 μs. Entretanto, em uma transferência de dados usando uma parcela de tempo, são necessários 259 μs para mecanismos de salto e de controle. Isto significa que a duração de uma parcela de tempo é de apenas 625 – 259, ou 366 μs. Com uma largura de banda de 1 MHz e 1 *bit*/Hz, o tamanho de um quadro, nesse caso, fica sendo de 366 *bits*.

Um quadro de três parcelas ocupa três parcelas de tempo. Entretanto, como 259 μs são usados para saltos, o tamanho do quadro é de 3 × 625 – 259 = 1.616 μs ou 1.616 *bits*. Um dispositivo que usa quadros de três parcelas permanece no mesmo salto (na mesma frequência portadora) durante três parcelas de tempo. Embora apenas um número de salto seja usado, três números de salto são consumidos. Isto significa que o número de salto de cada quadro é igual àquele presente na primeira parcela de tempo ocupada pelo quadro.

Um quadro de cinco parcelas também usa 259 *bits* para saltos, o que significa que o tamanho do quadro é de 5 × 625 – 259 = 2.866 *bits*.

A Figura 6.25 mostra o formato dos três tipos de quadros discutidos anteriormente.

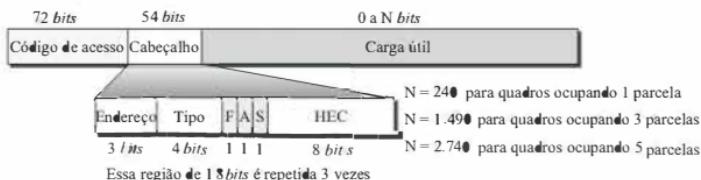



Figura 6.25 Tipos de formato de quadro.

Cada um dos campos listados na figura anterior é descrito a seguir:

- **Código de acesso.** Campo de 72 *bits* que normalmente contém *bits* de sincronização e o identificador da estação primária para que seja possível distinguir entre os quadros de diferentes *piconets*.
- **Cabeçalho.** Campo de 54 *bits* que é uma sequência de 18 *bits* repetidos três vezes. Cada sequência apresenta os seguintes subcampos:

  a. **Endereço.** O subcampo de endereço, composto por 3 *bits*, pode identificar até sete estações secundárias (1 a 7). O endereço de valor zero é usado para a comunicação via *broadcast*, da estação primária para todas as secundárias.
  b. **Tipo.** O subcampo de tipo, composto por 4 *bits*, especifica o tipo de dados proveniente das camadas superiores. Discutimos esses tipos mais adiante.
  c. **F.** Subcampo de 1 *bit*, usado para controle de fluxo. Quando ele vale 1, indica que o dispositivo não é capaz de receber mais quadros (o *buffer* está cheio).
  d. **A.** Subcampo de 1 *bit*, usado para confirmação de recebimento. O Bluetooth usa o método ARQ Stop-and-Wait, de modo que um único *bit* já é suficiente para enviar a confirmação.
  e. **S.** Subcampo de 1 *bit*, que contém um número de sequência. O Bluetooth usa o ARQ *Stop-and-Wait*, de modo que um único *bit* já é suficiente para numerar a sequência de quadros.
  f. **HEC.** O subcampo de Correção de Erros do Cabeçalho (HEC – Header Error Correction), de 8 *bits*, é uma soma de verificação que permite detectar erros em

cada seção de 18 *bits* do cabeçalho. O cabeçalho tem três seções idênticas de 18 *bits*. O receptor compara essas três seções, *bit* a *bit*. Se os *bits* nas posições correspondentes forem todos iguais, o *bit* é aceito; caso contrário, é aplicada a regra da maioria. Essa é uma forma de correção antecipada de erros (para o cabeçalho apenas). O controle duplo de erros é necessário porque a natureza da comunicação, através do ar, apresenta muito ruído. Perceba que não há retransmissão nessa subcamada.

- **Carga útil.** Subcampo que pode ter de 0 a 2.740 *bits* de comprimento. Ele contém dados ou informações de controle provenientes das camadas superiores.

Camada de rádio

A camada de rádio é aproximadamente equivalente à camada física do modelo Internet. Os dispositivos Bluetooth são equipamentos de baixa potência e têm um alcance de 10 m*.

Banda O Bluetooth utiliza a banda ISM de 2,4 GHz, que é dividida em 79 canais de 1 MHz cada.

FHSS O Bluetooth usa o método Espalhamento Espectral por Saltos em Frequências (FHSS – Frequency Hopping Spread Spectrum) na camada física para evitar a interferência de outros dispositivos ou de outras redes. O Bluetooth realiza 1.600 saltos por segundo, o que significa que cada dispositivo troca sua frequência de modulação 1.600 vezes por segundo. Um dispositivo usa uma frequência de apenas 625 μs (1/1.600 s) antes de saltar para outra; o tempo de permanência em cada frequência é de 625 μs.

Modulação Para transformar os *bits* em um sinal, o Bluetooth utiliza uma versão sofisticada do FSK, denominada GFSK (FSK com filtragem Gaussiana de banda; uma discussão sobre esse tema está fora do escopo deste livro). O GFSK tem uma frequência portadora. O *bit* 1 é representado por um aumento na frequência, que fica acima da portadora; o *bit* 0 é representado por uma redução na frequência, que fica abaixo da portadora. As frequências, em mega-hertz, são definidas de acordo com a seguinte fórmula para cada canal.

$$f_c = 2.402 + n \text{ MHz} \qquad n = 0, 1, 2, 3, \ldots, 78$$

Por exemplo, o primeiro canal utiliza uma frequência portadora de 2.402 MHz (2,402 GHz) e o segundo, de 2.403 MHz (2,403 GHz).

6.1.4 WiMAX

O **Interoperabilidade Mundial para Acesso por Micro-ondas** (WiMAX – Worldwide Interoperability for Microwave Access) é um padrão IEEE 802.16 (para redes sem fio fixas) e 802.16e (para redes sem fio móveis) que tem como objetivo fornecer a "última milha" do acesso de banda larga sem fio, oferecendo uma alternativa ao serviço de telefonia DSL via modem a cabo. O WiMAX oferece cobertura e vazão ótimas para assinantes em linha de visada (LOS – line-of-sight), ou seja, para os quais o caminho não esteja obstruído. Ele também oferece cobertura e vazão aceitáveis para assinantes que estejam próximos à estação-base ou sem linha de visada (NLOS – non-line-of-sight) em relação a ela.

Muitos usuários comparam o WiMAX ao WiFi. O WiMAX, assim como o WiFi, apresenta uma infraestrutura com uma estação de base, porém oferece muito mais do que o WiFi. Enquanto o WiFi só cobre uma região de cerca de 100 metros, o WiMAX tem um alcance de seis quilômetros. O WiMAX oferece substancialmente mais segurança, confiabilidade, qualidade de serviço e vazão do que o WiFi.

---

* N. de T.: As novas versões do Bluetooth especificam classes de potência onde são definidos dispositivos com alcance de até 100 metros.

Arquitetura

Discutimos brevemente a arquitetura do WiMAX nesta seção.

Estação-base

As unidades básicas de uma estação-base WiMAX são um rádio e uma antena. Cada rádio WiMAX tem tanto um transmissor quanto um receptor e transmite sinais com uma frequência entre 2 e 11 GHz. O WiMAX usa um sistema de Rádio Definido por *Software* (SDR – Software Defined Radio).

Três diferentes tipos de antenas (omnidirecional, setorial e painel) são usados no WiMAX para melhorar o desempenho de uma dada aplicação. O WiMAX usa um **Sistema de Antenas Adaptativas** (AAS – Adaptive Antenna System) com direcionamento do feixe de comunicação. Durante a transmissão, uma antena AAS pode focalizar sua energia de transmissão na direção de um receptor; durante a recepção, ela pode mover seu foco na direção do dispositivo emissor.

Outras medidas tomadas pelo WiMAX para evitar a interferência são a aplicação de OFDMA e de um sistema de antenas MIMO. O OFDMA é um método de acesso múltiplo que permite transmissões simultâneas de e para diversos usuários, funcionando bem em conjunto com AAS e MIMO para melhorar significativamente a vazão, aumentar o alcance e reduzir interferências.

Estações de assinantes

O **Equipamento nas Instalações do Cliente** (CPE – Customer Premises Equipment), também conhecido como *unidade do assinante* ou *equipamento terminal*, é disponibilizado em duas versões, para ambientes internos ou externos. A unidade interna tem o tamanho de um modem DSL ou a cabo. Ela pode ser instalada pelo cliente, porém requer que o assinante esteja mais perto de uma estação base devido às perdas da transmissão via rádio. A versão externa é do tamanho de uma antena parabólica residencial e deve ser instalada por um profissional.

Unidade portátil

Devido ao potencial do WiMAX móvel, há um foco crescente em unidades portáteis, que incluem aparelhos celulares, periféricos para PC, dispositivos embarcados em laptops e dispositivos eletrônicos (como consoles de videogame, leitores de MP3 e similares).

Camada de enlace de dados

A subcamada MAC no WiFi utiliza acesso com contenção. Isto pode fazer com que estações de assinantes distantes do AP sejam repetidamente interrompidas por estações mais próximas. A subcamada MAC no WiMAX usa um algoritmo de escalonamento. A estação do assinante deve competir com outras estações uma única vez, quando ela entra na rede. Uma parcela de tempo de acesso é então atribuída àquela estação de assinante.

Camada física

O 802.16e-2005 especifica o uso da faixa de 2 a 11 GHz, **OFDMA Escalável** (SOFDMA – Scalable OFDMA), uso de antenas MIMO e suporte total à mobilidade.

Aplicação

O WiMAX tem como objetivo prover alternativas economicamente viáveis a diversas infraestruturas de telecomunicação existentes, incluindo as redes de fio de cobre e celulares das empresas de telefonia e a infraestrutura de cabo coaxial das empresas de TV a cabo.

6.2 OUTRAS REDES SEM FIO

Nesta seção, nos concentramos em outras redes sem fio. Começamos discutindo a telefonia celular, que é uma tecnologia ubíqua (onipresente). Em seguida, cobrimos redes de satélites. Antes de discutirmos as redes sem fio citadas anteriormente, entretanto, abordaremos um método de acesso apenas mencionado no Capítulo 5, mas cuja descrição foi adiada para este: o método de canalização, usado em redes celulares e outras redes sem fio.

6.2.1 Canalização

A **canalização** (ou *partição de canal*, como também é conhecida) é um método de acesso múltiplo no qual a largura de banda disponível em um canal é compartilhada no tempo, na frequência ou por meio de um código, entre as diferentes estações. Nesta seção, discutimos três protocolos de canalização: FDMA, TDMA e CDMA.

Acesso Múltiplo por Divisão de Frequência

No **Acesso Múltiplo por Divisão de Frequência** (FDMA – Frequency-Division Multiple Access), a largura de banda disponível é dividida em faixas de frequência. Cada estação tem uma banda a ela atribuída para enviar seus dados. Em outras palavras, cada banda é reservada para uma estação específica e ela pertence àquela estação o tempo todo. Cada estação também usa um filtro passa-faixa para limitar as frequências de transmissão. Para evitar interferências entre as estações, as bandas atribuídas são separadas umas das outras por *bandas de guarda* de pequenas dimensões. A Figura 6.26 ilustra a ideia do FDMA.

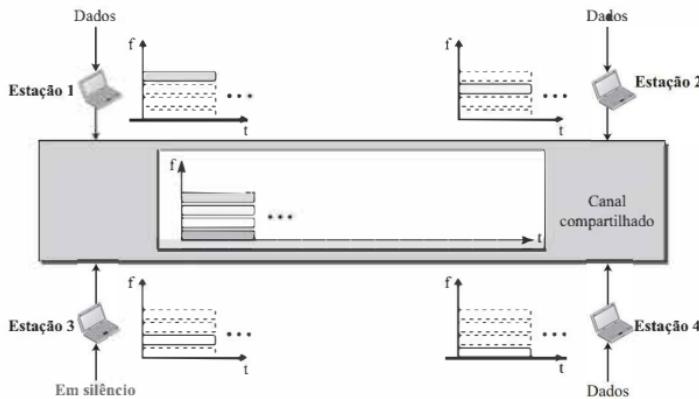



Figura 6.26 Acesso Múltiplo por Divisão de Frequência (FDMA).

O FDMA especifica uma banda de frequência predeterminada por toda a duração da comunicação. Isto significa que um fluxo de dados (um fluxo contínuo de dados que não pode ser empacotado) pode ser facilmente transmitido com o FDMA. Veremos mais adiante como esse recurso pode ser utilizado em sistemas de telefonia celular.

Precisamos enfatizar que o FDMA é um protocolo da camada de enlace; ele não deve ser confundido com um processo de multiplexação como a Multiplexação por Divisão de Frequência (FDM – Frequency-Division Multiplexing), que discutimos no Capítulo 7. A FDM é, na realidade, a implementação do FDMA na camada física.

Acesso Múltiplo por Divisão de Tempo

No **Acesso Múltiplo por Divisão de Tempo** (TDMA – Time-Division Multiple Access), as estações compartilham a largura de banda do canal ao longo do tempo. Cada estação tem uma banda a ela atribuída para enviar seus dados. A cada estação é atribuído um intervalo de tempo durante o qual ela pode enviar dados. Cada estação transmite os dados na parcela de tempo a ela atribuída. A Figura 6.27 mostra a ideia por trás do TDMA.

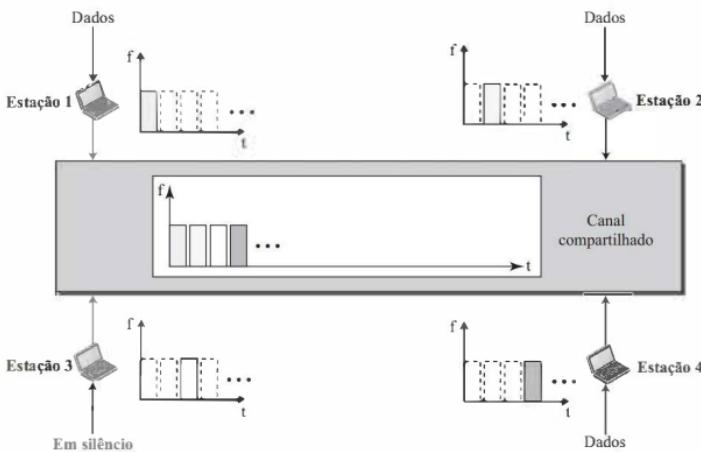



Figura 6.27 Acesso Múltiplo por Divisão de Tempo (TDMA).

O principal problema do TDMA refere-se à necessidade de sincronização entre as diferentes estações. Cada estação precisa saber quando sua parcela de tempo inicia e qual a localização dela. Isto pode ser difícil se as estações estiverem espalhadas em uma grande área, devido a atrasos de propagação introduzidos no sistema. Para compensar os atrasos, podemos inserir *tempos de guarda*. A sincronização é normalmente obtida usando alguns *bits* de sincronização (normalmente denominados *bits* de preâmbulo) no início de cada parcela de tempo.

Precisamos também enfatizar que, embora o TDMA e a Multiplexação por Divisão de Tempo (TDM – Time-Division Multiplexing) pareçam conceitualmente iguais, existem diferenças entre eles. A TDM, conforme discutido no Capítulo 7, é uma técnica da camada física que combina os dados provenientes de canais mais lentos e os transmite usando um canal mais rápido. O processo utiliza um multiplexador físico que intercala unidades de dados de cada canal.

O TDMA, por outro lado, é um método de acesso da camada de enlace de dados, que, em cada estação, pede à sua camada física que ela use a parcela de tempo alocada àquela estação. Não há um multiplexador físico na camada de enlace de dados.

Acesso Múltiplo por Divisão de Código

O **Acesso Múltiplo por Divisão de Código** (CDMA – Code-Division Multiple Access) foi concebido há várias décadas. Porém, foram os avanços recentes na tecnologia eletrônica que finalmente tornaram sua implementação possível. O CDMA difere do FDMA porque um único canal ocupa toda a largura de banda do enlace. Ele difere do TDMA porque todas as estações podem enviar dados simultaneamente; não há compartilhamento de tempo.

Analogia

Primeiramente, faremos uma analogia. O CDMA significa simplesmente uma comunicação com diferentes códigos. Por exemplo, em uma grande sala com diversas pessoas, duas podem falar em inglês se nenhuma outra pessoa na sala entender inglês. Outras duas podem falar em chinês caso elas sejam as únicas a entender chinês, e assim por diante. Em outras palavras, o canal compartilhado, que nesse caso é o espaço da sala, pode facilmente permitir a comunicação entre diversos pares de interlocutores se ela for feita usando diferentes idiomas (códigos).

Ideia

Consideremos que existam quatro estações, 1, 2, 3 e 4, conectadas ao mesmo canal. Os dados da estação 1 são denotados d_1, os da estação 2 são denotados d_2 e assim por diante. O código atribuído à primeira estação é c_1, à segunda estação é c_2 e assim por diante. Consideramos que os códigos atribuídos apresentam duas propriedades.

1. Se multiplicarmos cada código por outro, o resultado é 0.
2. Se multiplicarmos cada código por ele mesmo, o resultado é 4 (o número de estações).

Com essas duas propriedades em mente, vejamos como as quatro estações mencionadas anteriormente podem enviar dados usando o mesmo canal compartilhado, conforme mostra a Figura 6.28.

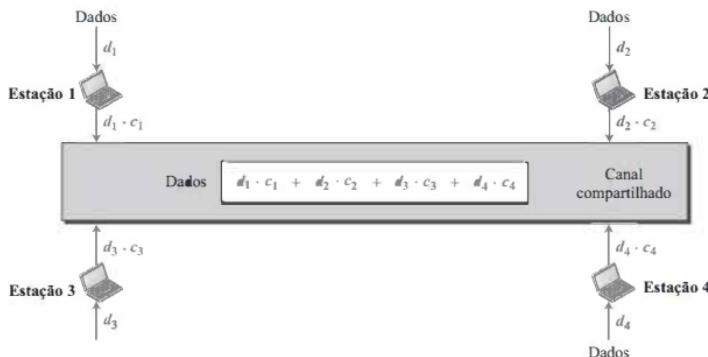



Figura 6.28 Ideia simples da comunicação com códigos.

A estação 1 multiplica (um tipo especial de multiplicação, conforme veremos) seus dados pelo seu código para obter $d_1 \cdot c_1$. A estação 2 multiplica seus dados pelo seu código para obter $d_2 \cdot c_2$ e assim por diante. Os dados colocados no canal correspondem à soma de todos esses termos, conforme mostrado na caixa no centro da figura. Qualquer estação que queira receber dados provenientes

de uma das outras três estações multiplica os dados no canal pelo código do emissor. Por exemplo, considere que as estações 1 e 2 estejam se comunicando. A estação 2 quer receber os dados que a estação 1 está enviando. A estação 2 multiplica os dados no canal por c_1, o código da estação 1.

Como o resultado de $(c_1 \cdot c_1)$ é 4, enquanto $(c_2 \cdot c_1)$, $(c_3 \cdot c_1)$ e $(c_4 \cdot c_1)$ resultam todos em 0s, a estação 2 divide o resultado por 4 para obter os dados provenientes da estação 1.

$$\text{dados} = [(d_1 \cdot c_1 + d_2 \cdot c_2 + d_3 \cdot c_3 + d_4 \cdot c_4) \cdot c_1] / 4$$
$$= [d_1 \cdot c_1 \cdot c_1 + d_2 \cdot c_2 \cdot c_1 + d_3 \cdot c_3 \cdot c_1 + d_4 \cdot c_4 \cdot c_1] / 4 = (4 \times d_1) / 4 = d_1$$

Fichas

O CDMA se baseia na teoria de códigos. A cada estação é atribuído um código, que consiste em uma sequência de números conhecidos como fichas ou *chips*. A Figura 6.29 mostra as fichas para o exemplo anterior.

Figura 6.29 Sequências de fichas.

Ainda neste capítulo mostraremos como os códigos são selecionados. Por ora, precisamos saber que não escolhemos aleatoriamente as sequências; elas foram cuidadosamente selecionadas. São denominadas **sequências ortogonais** e apresentam as seguintes propriedades:

1. Cada sequência é composta por N elementos, onde N é o número de estações e precisa ser uma potência de 2.
2. Se multiplicarmos uma sequência por um número, todos os elementos da sequência são multiplicados por aquele número. Isto é chamado de multiplicação de uma sequência por um escalar. Por exemplo,

   2 • [+1 +1 −1 −1] = [+2 +2 −2 +2].

3. Se multiplicarmos duas sequências iguais, elemento por elemento, e somarmos os resultados, obtemos N, onde N é o número de elementos em cada sequência. Isto é chamado *produto interno* de duas sequências iguais. Por exemplo,

   [+1 +1 −1 −1] • [+1 +1 −1 −1] = 1 + 1 + 1 + 1 = 4.

4. Se multiplicarmos duas sequências distintas, elemento por elemento, e somarmos os resultados, obtemos 0. Isto é chamado produto interno de duas sequências distintas. Por exemplo,

   [+1 +1 −1 −1] • [+1 +1 +1 +1] = 1 + 1 − 1 − 1 = 0.

5. Somar duas sequências significa somar os elementos correspondentes. O resultado é outra sequência. Por exemplo,

   [+1 +1 −1 −1] + [+1 +1 +1 +1] = [+2 +2 0 0].

Representação dos dados

Seguimos algumas regras para a codificação: se uma estação precisar enviar o *bit* 0, ela codifica esse *bit* como −1; se precisar enviar o *bit* 1, ela codifica esse *bit* como +1. Quando uma estação estiver ociosa, nenhum sinal é enviado por ela, o que é interpretado como um 0. Essas regras são mostradas na Figura 6.30.

Figura 6.30 Representação dos dados no CDMA.

Codificação e decodificação

Como um exemplo simples, mostramos como quatro estações compartilham o canal durante um intervalo de 1 *bit*. O procedimento pode ser facilmente repetido para intervalos adicionais. Consideramos que as estações 1 e 2 estão enviando um *bit* 0 e a estação 4 está enviando um *bit* 1. A estação 3 está em silêncio. Os dados no lado do emissor são traduzidos para −1, −1, 0 e +1. Cada estação multiplica o número correspondente pela sua ficha (a sua sequência ortogonal), que é única para cada estação. O resultado é uma nova sequência que é enviada ao canal. Por razões de simplicidade, consideramos que todas as estações enviam as sequências resultantes ao mesmo tempo. A sequência no canal é a soma de todas as quatro sequências, conforme definido anteriormente. A Figura 6.31 ilustra essa situação.

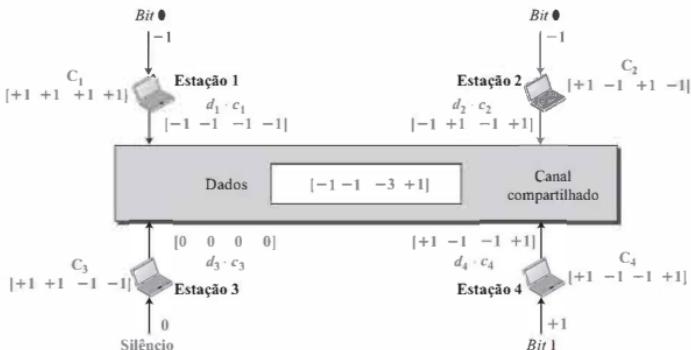



Figura 6.31 Compartilhando o canal no CDMA.

Agora, considere que a estação 3, que dissemos estar em silêncio, deseja recuperar os dados enviados pela estação 2. A estação 3 multiplica o total dos dados no canal pelo código da estação 2, que é [+1 −1 +1 −1] para obter

$$[-1\ -1\ -3\ +1] \cdot [+1\ -1\ +1\ -1] = -4$$
$$-4/4 = -1 \rightarrow \textbf{\textit{bit 0}}.$$

Nível do sinal

O processo pode ser melhor compreendido se mostrarmos o sinal digital produzido por cada estação e os dados recuperados no destino (ver Figura 6.32). A Figura mostra os sinais correspondentes a cada estação usando um sinal NRZ-L (ver Capítulo 7) e o sinal resultante no canal compartilhado.

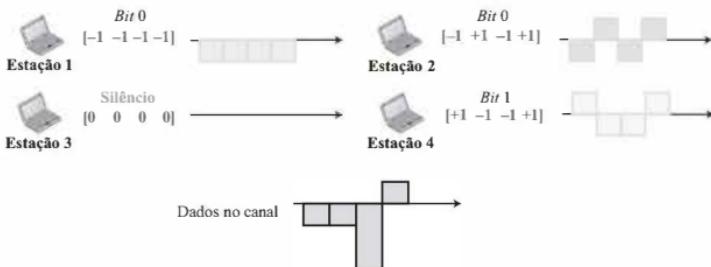



Figura 6.32 Sinal digital criado por quatro estações no CDMA.

A Figura 6.33 mostra como a estação 3 pode detectar os dados enviados pela estação 2 usando o código atribuído a esta. Os dados presentes no canal são multiplicados (operação de produto interno) pelo sinal que representa o código da ficha da estação 2 para obter um novo sinal. A estação, então, integra e soma a área sob o sinal, obtendo o valor -4, que é dividido por 4 e interpretado como um *bit* 0.

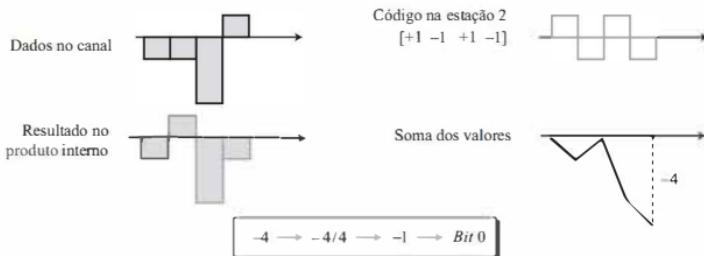



Figura 6.33 Decodificação do sinal composto no CDMA.

Geração da sequência

Para gerar sequências de fichas, usamos uma tabela de Walsh, uma tabela bidimensional com um número igual de linhas e colunas, conforme mostrado na Figura 6.34. Na tabela de Walsh, cada linha corresponde a uma sequência de *chips*. A tabela W_1 para uma sequência de um *chip* tem uma linha e uma coluna. Podemos escolher –1 ou +1 para a ficha nessa tabela trivial (escolhemos +1). Segundo Walsh, se conhecermos a tabela para *N* sequências, denotada W_N, podemos construir a tabela para 2*N* sequências, denotada W_{2N}, conforme mostra a Figura 6.34. Escrevemos W_N com uma barra sobre ela para representar o complemento de W_N, no qual cada +1 é alterado para –1 e vice-

$$W_1 = \begin{bmatrix} +1 \end{bmatrix} \quad W_{2N} = \begin{bmatrix} W_N & W_N \\ W_N & \overline{W_N} \end{bmatrix}$$

(*a*) Duas regras básicas

$$W_2 = \begin{bmatrix} +1 & +1 \\ +1 & -1 \end{bmatrix} \quad W_4 = \begin{bmatrix} +1 & +1 & +1 & +1 \\ +1 & -1 & +1 & -1 \\ +1 & +1 & -1 & -1 \\ +1 & -1 & -1 & +1 \end{bmatrix}$$

(*b*) Geração de W_1, W_2, e W_4

Figura 6.34 Regra geral e exemplos da criação de tabelas de Walsh.

-versa. A Figura 6.34 também mostra como podemos construir W_2 e W_4 a partir de W_1. Depois de selecionarmos W_1, W_2 pode ser construída a partir de quatro W_1, sendo o último elemento o complemento de W_1. Após gerar W_2, podemos construir W_4 a partir de quatro W_2, sendo o último elemento o complemento de W_2. Obviamente, W_8 é composto por quatro W_4 e assim por diante. Perceba que, após a tabela W_N ser construída, é atribuído a cada estação uma ficha correspondente a uma linha.

Algo que precisamos enfatizar é que o número de sequências N precisa ser uma potência de 2. Em outras palavras, precisamos ter $N = 2^m$.

Exemplo 6.1

Determine as fichas para uma rede com

a. Duas estações
b. Quatro estações

Solução

Podemos usar as linhas de W_2 e W_4 mostradas na Figura 6.34:

a. Para uma rede com duas estações, temos [+1 +1] e [+1 −1].
b. Para uma rede com quatro estações, temos [+1 +1 +1 +1], [+1 −1 +1 −1], [+1 +1 −1 −1] e [+1 −1 −1 +1].

Exemplo 6.2

Qual deve ser o número de sequências se tivermos 90 estações em nossa rede?

Solução

O número de sequências N precisa satisfazer $N = 2^m$. Precisamos escolher $m = 7$ e $N = 2^7$ ou 128. Podemos, então, usar 90 das sequências possíveis como fichas.

Exemplo 6.3

Mostre que uma estação receptora pode obter os dados enviados por um emissor específico se ela multiplicar todos os dados no canal pela ficha referente ao código do emissor e depois dividir o resultado pelo número de estações.

Solução

Mostraremos isto para a primeira estação, usando o nosso exemplo anterior de quatro estações. Podemos escrever os dados do canal como $D = (d_1 \cdot c_1 + d_2 \cdot c_2 + d_3 \cdot c_3 + d_4 \cdot c_4)$. O receptor, que deseja obter os dados enviados pela estação 1, multiplica esses dados por c_1.

$$
\begin{aligned}
[D \cdot c_1] / 4 &= [(d_1 \cdot c_1 + d_2 \cdot c_2 + d_3 \cdot c_3 + d_4 \cdot c_4) \cdot c_1] / 4 \\
&= [\, d_1 \cdot c_1 \cdot c_1 + d_2 \cdot c_2 \cdot c_1 + d_3 \cdot c_3 \cdot c_1 + d_4 \cdot c_4 \cdot c_1] / 4 \\
&= [\, d_1 \times 4 + d_2 \times 0 + d_3 \times 0 + d_4 \times 0] / 4 = [d_1 \times 4] / 4 = d_1
\end{aligned}
$$

6.2.2 Telefonia celular

A **telefonia celular** foi projetada para fornecer comunicação entre duas unidades em movimento, denominadas *estações móveis* (EMs), ou entre uma unidade móvel e uma unidade estacionária, comumente denominada *estação terrestre*. Um provedor de serviços deve ser capaz de localizar e rastrear uma chamada, atribuir um canal para ela e transferir o canal de uma estação-base para outra à medida que uma estação móvel sai da área de cobertura da estação-base atual.

Para que o rastreamento seja possível, cada área do serviço de telefonia celular é dividida em pequenas regiões denominadas *células*. Cada célula contém uma antena e é controlada por uma estação alimentada por energia solar ou elétrica, denominada *estação-base* (EB). Cada estação-base, por sua vez, é controlada por uma unidade de comutação conhecida como **Central de Comutação Móvel** (MSC – Mobile Switching Center). A MSC coordena a comunicação entre todas as estações-base e a central telefônica*. Ela consiste em um centro informatizado responsável por conectar as chamadas, por gravar informações da chamada e pela tarifação (ver Figura 6.35).

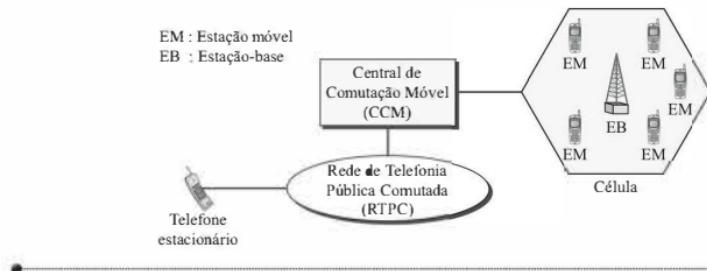



Figura 6.35 Sistema de telefonia celular.

O tamanho da célula não é fixo e pode ser aumentado ou diminuído dependendo da população na área. O raio típico de uma célula é de 1 a 20 Km. Para atender a demanda por tráfego, áreas com alta densidade populacional exigem um número maior de células geograficamente menores do que áreas de baixa densidade. Uma vez determinado, o tamanho da célula é ajustado para evitar interferências de sinais provenientes das células adjacentes. A potência de transmissão de cada célula é mantida baixa para evitar que seu sinal interfira com os sinais de outras células.

Princípio do reuso de frequências

Em geral, células vizinhas não podem usar o mesmo conjunto de frequências para a comunicação porque podem ocorrer interferências para os usuários em locais próximos aos limites da célula. Entretanto, o conjunto de frequências disponíveis é limitado, de forma que elas devem ser reutilizadas. Um padrão de reuso de frequências consiste em uma configuração de N células, sendo N o **fator de**

* N. de T.: A MSC também é responsável pela interligação da rede de telefonia celular com o sistema de telefonia tradicional, conhecido como Rede de Telefonia Pública Comutada (PSTN – Public Switched Telephone Network).

reuso, em que cada célula utiliza um conjunto único de frequências. Quando o padrão se repete, as frequências podem ser reutilizadas. Existem diversos padrões diferentes possíveis. A Figura 6.36 mostra dois deles.

(*a*) Fator de reuso de 4 (*b*) Fator de reuso de 7

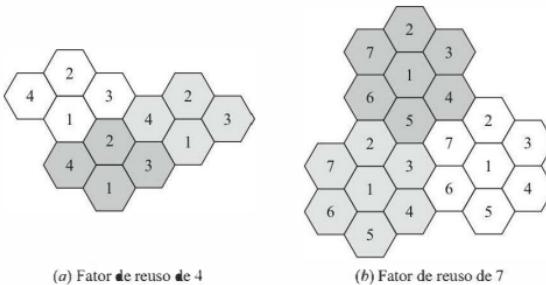



Figura 6.36 Padrão de reutilização de frequências.

Os números nas células definem o padrão. As células com o mesmo número em um padrão podem usar o mesmo conjunto de frequências. Chamamos essas células de *células de reuso*. Conforme mostra a Figura 6.36, em um padrão com um fator de reuso de 4, há apenas uma célula separando as células que usam o mesmo conjunto de frequências. Em um padrão com fator de reuso de 7, há duas células separando as células de reuso.

Transmissão

Para fazer uma chamada a partir de uma estação móvel, a pessoa que faz a chamada insere um código de 7 ou 10 dígitos (um número de telefone) e pressiona o botão de envio. A estação móvel, então, examina a banda, buscando um canal com um sinal suficientemente forte, e então envia os dados (número de telefone) para a estação-base mais próxima usando tal canal. A estação-base encaminha os dados para a MSC, que envia os dados para a central telefônica. Se o número chamado estiver disponível, uma conexão é criada e o resultado é encaminhado de volta para a MSC. Em seguida, a MSC atribui a esta chamada um canal de voz que não esteja sendo utilizado e uma conexão é estabelecida. A estação móvel ajusta automaticamente sua frequência para o novo canal e a comunicação pode começar.

Recepção

Quando um telefone móvel é chamado, a central telefônica envia o número para a MSC, que busca a localização da estação móvel correspondente, enviando sinais de consultas a cada célula em um processo conhecido como *paging*. Assim que a estação móvel é encontrada, a MSC transmite um sinal de campainha (o toque do telefone) e, quando a estação móvel responde, ela atribui um canal de voz para a chamada, permitindo que a comunicação de voz se inicie.

Transferência

É possível que, durante uma conversa, a estação móvel se mova de uma célula para outra. Quando isso acontece, o sinal pode ficar fraco. Para resolver esse problema, a MSC monitora o nível do sinal em intervalos de poucos segundos. Se a intensidade do sinal diminuir, a MSC procura uma nova célula que possa acomodar melhor a comunicação. A MSC muda então o canal transportando a chamada (transfere o sinal controlado pelo canal antigo para um novo canal).

Hard handoff Os sistemas de telefonia iniciais adotavam o chamado ***hard handoff***, também conhecido como *transferência rígida*. Em um *hard handoff*, cada estação móvel se comunica com apenas uma estação-base a cada instante. Quando a EM se move de uma célula para outra, a comunicação com a estação-base anterior deve ser interrompida antes que a comunicação com a nova estação-base possa ser estabelecida. Isto pode criar uma transição abrupta.

Soft handoff Sistemas mais novos adotam o chamado ***soft handoff***, ou *transferência suave*. Nesse caso, cada estação móvel pode se comunicar com duas estações-base ao mesmo tempo. Isto significa que, durante o processo de *handoff*, uma estação móvel pode se comunicar com a nova estação-base antes de interromper a comunicação com a antiga.

Roaming

Uma característica da telefonia celular é o chamado ***roaming***, algumas vezes também denominado *itinerância*. *Roaming* significa, em princípio, que um usuário pode ter acesso à rede de comunicação ou pode ser chamado onde houver cobertura. Um provedor de serviços geralmente tem cobertura limitada. Provedores de serviços vizinhos podem fornecer uma cobertura estendida por meio de um contrato de *roaming*. A situação é semelhante ao correio tradicional entre países. A taxa de entrega de uma carta entre dois países pode ser dividida conforme combinado entre os dois países.

Primeira geração (1G)

A telefonia celular encontra-se atualmente em sua quarta geração. A primeira geração foi projetada para a comunicação de voz usando sinais analógicos. Discutimos um sistema de primeira geração móvel utilizado na América do Norte, o AMPS.

AMPS

O **Sistema Avançado de Telefonia Móvel** (AMPS – Advanced Mobile Phone System) é um dos principais sistemas de telefonia celular analógica na América do Norte. Ele usa FDMA (ver Capítulo 5) para separar os canais em uma conexão.

O AMPS é um sistema de telefonia celular analógica que usa FDMA.

Bandas O AMPS opera na banda ISM de 800 MHz. O sistema utiliza dois canais analógicos separados, um para a comunicação direta (estação-base para estação móvel) e outro para a comunicação reversa (estação móvel para estação-base). A banda entre 824 e 849 MHz transporta a comunicação reversa; a banda entre 869 e 894 MHz transporta a comunicação direta (ver Figura 6.37).

Figura 6.37 Bandas de telefonia celular no AMPS.

Cada banda é dividida em 832 canais. Entretanto, dois provedores podem compartilhar uma área, o que significa que cada provedor usa 416 canais de cada célula. Dos 416 canais, 21 são usados para controle, o que leva a 395 canais restantes. O AMPS tem um fator de reuso de frequências de 7, o que significa que apenas um sétimo desses 395 canais de tráfego ficam de fato disponíveis em uma célula.

Transmissão O AMPS usa FM e FSK para a modulação. A Figura 6.38 mostra a transmissão no sentido reverso. Canais de voz são modulados usando FM, enquanto canais de controle usam FSK para criar sinais analógicos de 30 kHz. O AMPS utiliza FDMA para dividir cada banda de 25 MHz em canais de 30 kHz.

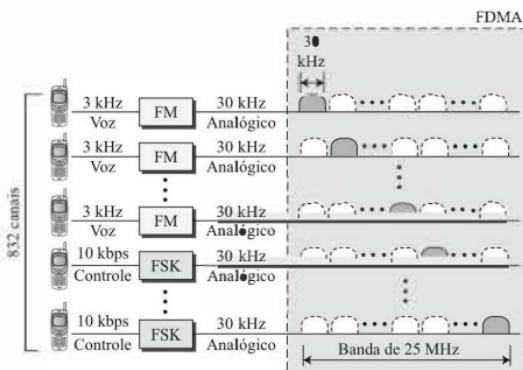



Figura 6.38 Banda de comunicação reversa do AMPS.

Segunda geração (2G)

A segunda geração da rede de telefonia celular foi desenvolvida para proporcionar maior qualidade (menor propensão a ruído) às comunicações móveis de voz. Enquanto a primeira geração foi projetada para comunicação de voz analógica, a segunda foi projetada principalmente para voz digitalizada. Três sistemas principais evoluíram na segunda geração: D-AMPS, GSM e CDMA.

D-AMPS

O produto da evolução do AMPS analógico em um sistema digital é o **AMPS digital** (D-AMPS). O D-AMPS foi projetado para ser retrocompatível com o AMPS. Isto significa que, em uma célula, um telefone pode usar AMPS e outro D-AMPS. O D-AMPS foi definido pela primeira vez no Padrão Provisório 54(IS-54 – Interim Standard 54), sendo revisado mais tarde no IS-136.

Banda O D-AMPS utiliza as mesmas bandas e canais que o AMPS.

Transmissão Cada canal de voz é digitalizado usando técnicas de PCM e de compressão muito complexas. Um canal de voz é digitalizado a 7,95 kbps. Três canais de voz digital de 7,95 kbps são combinados usando TDMA. O resultado são 48,6 kbps de dados digitais; porém, uma grande parte dessa taxa é ocupada por dados de controle e por redundância. Conforme mostra a Figura 6.39, o sistema envia 25 quadros por segundo, com 1.944 *bits* por quadro. Cada quadro tem uma duração de 40 ms (1/25) e é dividido em seis parcelas de tempo (*slots*) compartilhadas por três canais digitais; a cada canal são atribuídas duas dessas parcelas.

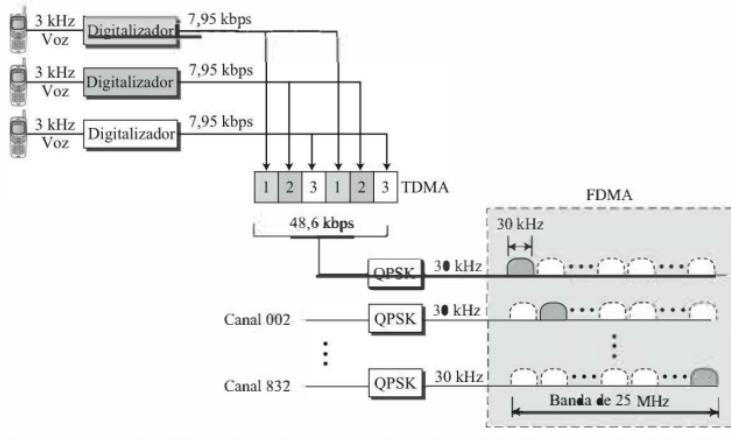



Figura 6.39 D-AMPS

Cada parcela de tempo comporta 324 *bits*. Entretanto, apenas 159 *bits* são provenientes da voz digitalizada; 64 *bits* são utilizados para controle e 101 *bits* são usados para correção de erros. Em outras palavras, cada canal alimenta 159 *bits* de dados úteis em cada um dos dois canais a ele atribuídos. O sistema adiciona 64 *bits* de controle e 101 *bits* para correção de erros.

Os 48,6 kbps de dados digitais resultantes modulam uma portadora utilizando QPSK; o resultado é um sinal analógico de 30 kHz. Finalmente, os sinais analógicos de 30 kHz compartilham uma banda de 25 MHz (FDMA). O D-AMPS adota um fator de reuso de frequências de 7.

O D-AMPS, ou IS-136, é um sistema de telefonia celular digital que usa TDMA e FDMA.

GSM

O **Sistema Global para Comunicações Móveis** (GSM – Global System for Mobile Communication) é um padrão europeu desenvolvido para fornecer uma tecnologia de segunda geração em comum para toda a Europa*. O objetivo era substituir as diversas tecnologias de primeira geração incompatíveis.

Bandas O GSM utiliza duas bandas para comunicação duplex. Cada banda tem uma largura de 25 MHz, sendo deslocada na direção dos 900 MHz conforme mostra a Figura 6.40. Cada banda é dividida em 124 canais de 200 kHz, separados por bandas de guarda.

Transmissão A Figura 6.41 mostra um sistema GSM. Cada canal de voz é digitalizado e comprimido para um sinal digital de 13 kbps. Cada parcela discreta de tempo transporta 156,25 *bits*. Cada quadro é compartilhado por oito parcelas de tempo (TDMA). Vinte e seis quadros também formam um multiquadro compartilhado (TDMA). Podemos calcular a taxa de *bits* de cada canal como se segue.

* N. de T.: Apesar de inicialmente restrito à Europa, o GSM está presente atualmente em mais de 200 países do mundo, incluindo o Brasil. Mais informações sobre o GSM podem ser obtidas em http://www.gsma.com.

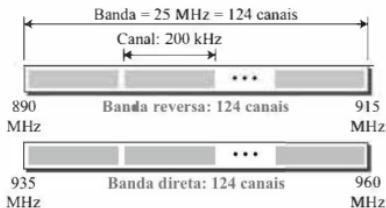



Figura 6.40 Bandas GSM.

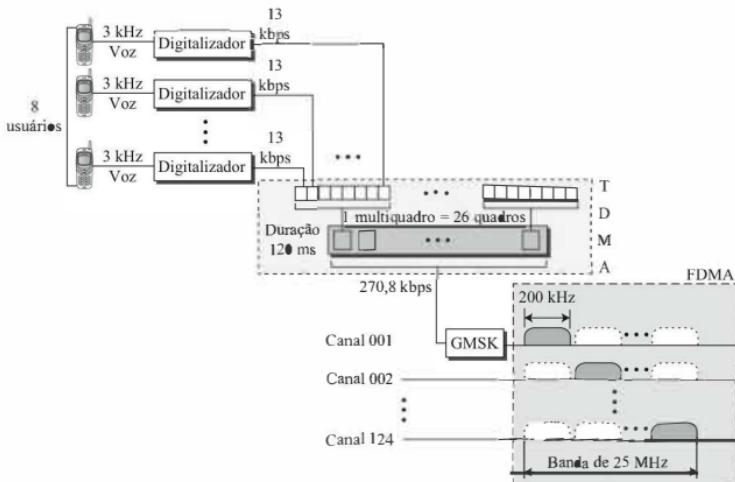



Figura 6.41 GSM.

Taxa de transferência de dados do canal = (1/120 ms) × 26 × 8 × 156,25 = 270,8 kbps

Cada canal digital de 270,8 kbps modula uma portadora usando GMSK (uma forma de FSK utilizada principalmente em sistemas europeus); o resultado é um sinal analógico de 200 kHz. Finalmente, 124 canais analógicos de 200 kHz são combinados usando FDMA. O resultado é uma banda de 25 MHz. A Figura 6.42 mostra os dados do usuário e de controle em um multiquadro.

O leitor deve ter notado a grande quantidade de carga de controle no TDMA. Os dados do usuário ocupam apenas 65 *bits* por parcela de tempo. O sistema adiciona *bits* extras para correção de erros de modo que cada parcela de tempo apresenta 114 *bits*. Além disso, são adicionados *bits* de controle para atingir 156,25 *bits* por parcela de tempo. Oito delas são encapsuladas em um quadro. Vinte e quatro quadros de tráfego e dois quadros extras de controle compõem um multiquadro, que tem duração de 120 ms. No entanto, a arquitetura GSM define superquadros e hiperquadros que não acrescentam qualquer carga de controle; não os discutiremos aqui.

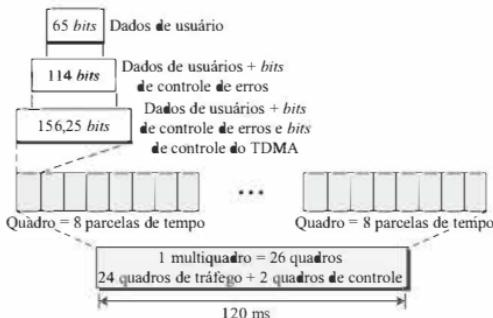



Figura 6.42 Componentes de um multiquadro.

Fator de reuso Devido ao seu complexo mecanismo de correção de erros, o GSM permite a adoção de um fator de reuso tão baixo quanto 3.

O GSM é um sistema de telefonia celular digital que usa TDMA e FDMA.

IS-95

Um dos padrões de segunda geração dominantes na América do Norte é o Padrão Provisório 95 (IS-95 – Interim Standard 95). Ele tem como base o CDMA e o DSSS.

Bandas e canais O IS-95 utiliza duas bandas para prover comunicação duplex. Elas podem ser a tradicional banda ISM de 800 MHz ou a banda ISM de 1.900 MHz. Cada banda é dividida em 20 canais de 1,228 MHz, separados por bandas de guarda. Cada provedor de serviços tem 10 canais a ele alocados. O IS-95 pode ser usado paralelamente ao AMPS. Cada canal IS-95 é equivalente a 41 canaisAMPS (41 × 30 kHz = 1,23 MHz).

Sincronização Todos os canais-base precisam ser sincronizados para usar o CDMA. Para se sincronizar, as estações base utilizam os serviços do Sistema de Posicionamento Global (GPS – Global Positioning System), um sistema de satélites que será discutido na próxima seção.

Transmissão direta O IS-95 apresenta duas técnicas de transmissão distintas: uma para uso no sentido direto (estação-base para dispositivo móvel) e outro para uso no sentido reverso (dispositivo móvel para estação-base). Na transmissão direta, as comunicações entre a estação-base e todos os equipamentos móveis são sincronizadas; a estação-base envia dados sincronizados para todos os dispositivos móveis. A Figura 6.43 mostra um diagrama simplificado no sentido direto.

Cada canal de voz é digitalizado, produzindo dados a uma taxa básica de 9,6 kbps. Após a adição dos *bits* de correção de erros e de repetição, bem como do intercalamento, o resultado obtido é um sinal de 19,2 ksps (milhares de sinais por segundo). Esses dados de saída são, então, embaralhados usando um sinal de 19,2 ksps. O sinal de embaralhamento é produzido a partir de um gerador de códigos longos que utiliza o Número de Série Eletrônico (ESN – Electronic Serial Number) da estação móvel e gera 2^{42} fichas pseudoaleatórias, cada ficha tendo 42 *bits*. Perceba que as fichas são geradas pseudoaleatoriamente, não aleatoriamente de fato, pois a sequência de fichas se repete. A saída do gerador de códigos longos é passada para um dispositivo conhecido

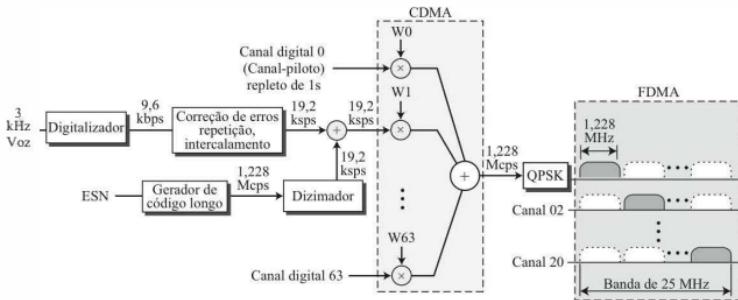



Figura 6.43 IS-95: transmissão no sentido direto.

como dizimador (*decimator*), que seleciona 1 *bit* a cada 64 *bits*. A saída do dizimador é utilizada no embaralhamento. O embaralhamento é usado para prover privacidade à comunicação; o ESN é único para cada estação.

O resultado do embaralhamento é combinado usando CDMA. Para cada canal de tráfego, uma ficha correspondente a uma linha de uma tabela de Walsh 64 × 64 é selecionada. O resultado é um sinal de 1,228 Mcps (*megachips* por segundo).

19,2 ksps × 64 cps = 1,228 Mcps

O sinal é passado para um modulador QPSK, que produz um sinal de 1,228 MHz. A largura de banda resultante é deslocada de forma adequada, usando FDMA. A partir de um canal analógico, são criados 64 canais digitais dos quais 55 são canais de tráfego (transportam voz digitalizada). Nove canais são utilizados para controle e sincronização:

a. O canal 0 é um canal-piloto. Ele envia um fluxo contínuo de *bits* 1 para as estações móveis. O fluxo permite a sincronização de *bits*, serve como uma referência de fase para a demodulação e permite que a estação móvel compare a potência do sinal de estações-base para tomar decisões relativas ao *handoff*.

b. O canal 32 fornece informações sobre o sistema para a estação móvel.

c. Os canais de 1 a 7 são usados para mecanismos de *paging*, permitindo o envio das mensagens para uma ou mais estações móveis.

d. Os canais de 8 a 31 e de 33 a 63 são canais de tráfego que transportam voz digitalizada da estação-base até a estação móvel correspondente.

Transmissão reversa O uso do CDMA no sentido direto é possível porque o canal-piloto envia uma sequência contínua de *bits* 1 para sincronizar a transmissão. A sincronização não é feita no sentido reverso porque isto exigiria uma entidade capaz de fazê-lo, o que não é viável. Em vez do CDMA, os canais reversos usam o Espalhamento Espectral por Sequência Direta (DSSS – Direct Sequence Spread Spectrum), discutido no Capítulo 7. A Figura 6.44 mostra um diagrama simplificado para a transmissão reversa.

Cada canal de voz é digitalizado, produzindo dados a uma taxa de 9,6 kbps. Entretanto, após a adição dos *bits* de correção de erros e de repetição, além do intercalamento, o resultado é um sinal de 28,8 ksps. Em seguida, a saída passa por um modulador de 6/64 símbolos. Os símbolos são divididos em blocos de seis símbolos e cada bloco é interpretado como um número binário (de 0 a 63). O

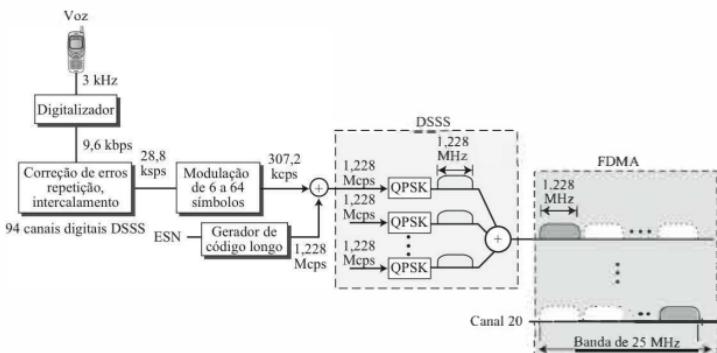



Figura 6.44 IS-95: transmissão no sentido reverso.

número binário é utilizado para indexar uma matriz de Walsh 64×64, levando à seleção de uma linha de fichas. Perceba que esse procedimento não é equivalente ao CDMA; os *bits* não são multiplicados pelas fichas em uma linha. Cada bloco de seis símbolos é substituído por um código de 64 fichas. Isso é feito para fornecer uma espécie de ortogonalidade, diferenciando os fluxos de *chips* das diferentes estações móveis. O resultado desse processo é a criação de um sinal de (28,8/6) × 64 ou 307,2 kcps.

O próximo passo consiste no espalhamento espectral; cada ficha é espalhada em 4. Novamente, o ESN da estação móvel é usado para gerar um código longo de 42 *bits* a uma taxa de 1,228 Mcps, que é o resultado de 4 vezes 307,2. Após o espalhamento, cada sinal é modulado usando um método de QPSK, o qual é ligeiramente diferente daquele utilizado no sentido direto; não entramos em detalhes aqui. Perceba que não há um mecanismo de acesso múltiplo aqui; todos os canais reversos enviam os seus sinais analógicos para o ar, porém as fichas corretas serão recebidas pela estação-base devido ao espalhamento espectral.

Embora possamos criar $2^{42} - 1$ canais digitais na direção reversa (devido ao gerador de código longo), normalmente são utilizados 94 canais; 62 são canais de tráfego, enquanto 32 são canais usados para ganhar acesso à estação-base.

O IS-95 é um sistema de telefonia celular digital que usa CDMA/DSSS e FDMA.

Dois conjuntos de taxa de transferência de dados O IS-95 define dois conjuntos de taxa de transferência de dados, com quatro taxas distintas em cada conjunto. O primeiro conjunto especifica 9.600, 4.800, 2.400 e 1.200 bps. Se, por exemplo, a taxa selecionada for de 1.200 bps, cada *bit* é repetido oito vezes para proporcionar uma taxa de 9.600 bps. O segundo conjunto define 14.400, 7.200, 3.600 e 1.800 bps. Isto é possível reduzindo-se o número de *bits* utilizados para correção de erros. As taxas de *bits* de um conjunto estão relacionadas com a atividade do canal. Se o canal estiver silencioso, apenas 1.200 *bits* podem ser transferidos, o que melhora o espalhamento devido à repetição de cada *bit* oito vezes.

Fator de reuso de frequências Em um sistema IS-95, o fator de reuso de frequências normalmente vale 1 porque a interferência causada por células vizinhas não é capaz de afetar uma transmissão CDMA ou DSSS.

Soft handoff Todas as estações-base fazem o *broadcast* continuo de sinais usando o seu canal-piloto. Isto significa que uma estação móvel pode detectar o sinal-piloto proveniente de sua célula e das células vizinhas. Isso permite que uma estação móvel faça um *soft handoff* em vez de um *hard handoff*.

Terceira geração (3G)

A terceira geração da telefonia celular consiste em uma combinação de tecnologias que fornecem comunicação tanto de dados digitais como de voz. Usando um pequeno dispositivo portátil, uma pessoa é capaz de falar com qualquer outra pessoa do mundo com uma qualidade de voz semelhante à da rede de telefonia fixa existente. Uma pessoa pode obter e assistir um filme, obter e ouvir músicas, navegar na Internet ou jogar em rede, fazer uma videoconferência e muito mais. Uma das características interessantes de um sistema de terceira geração é que o dispositivo portátil está sempre conectado; não é necessário discar um número para se conectar à Internet.

O conceito de terceira geração teve inicio em 1992, quando a ITU publicou um projeto denominado **Comunicação para Internet Móvel 2000** (IMT-2000 – Internet Mobile Communication 2000). O projeto especifica alguns critérios para a tecnologia de terceira geração, descritos a seguir:

a. Qualidade de voz comparável à da rede de telefonia pública existente.

b. Taxa de transferência de dados de 144 kbps para acesso a partir de veículos em movimento (automóveis), 384 kbps para acesso enquanto os usuários caminham (pedestres) e 2 Mbps para usuários que não estejam em movimento (no escritório ou em casa).

c. Suporte a serviços de dados baseados em comutação de pacotes e em comutação de circuitos.

d. Uma banda total de 2 GHz.

e. Larguras de banda de 2 MHz.

f. Interface com a Internet.

O principal objetivo da terceira geração de telefonia celular é fornecer comunicação pessoal universal.

Interface de rádio do IMT-2000

A Figura 6.45 mostra as interfaces de rádio (padrões sem fio) adotadas pelo IMT-2000. Todos os cinco foram desenvolvidos a partir de tecnologias de segunda geração. Os dois primeiros evoluíram a partir da tecnologia CDMA. O terceiro evoluiu a partir de uma combinação de CDMA e TDMA. A quarta é uma evolução do TDMA e a última evoluiu a partir do FDMA e do TDMA.

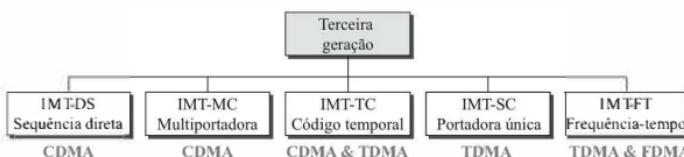



Figura 6.45 IMT-2000: interfaces de rádio.

IMT-DS Essa abordagem utiliza uma versão do CDMA denominada CDMA de Banda Larga (W-CDMA – Wideband CDMA). O W-CDMA utiliza uma largura de banda de 5 MHz. Ele foi desenvolvido na Europa e é compatível com o CDMA usado no IS-95.

IMT-MC Essa abordagem foi desenvolvida na América do Norte e é conhecida como CDMA 2000. Trata-se de uma evolução da tecnologia CDMA utilizada em canais IS-95. O CDMA 2000 combina espalhamento espectral da nova banda larga (15 MHz) com o CDMA de banda estreita (1,25 MHz) do IS-95. Ele é retrocompatível com o IS-95, permitindo a comunicação em múltiplos canais de 1,25 MHz (1, 3, 6, 9, 12 vezes), chegando a até 15 MHz. O uso dos canais mais largos permite que o CDMA 2000 atinja a taxa de transferência de dados de 2 Mbps especificada para a terceira geração.

IMT-TC Esse padrão utiliza uma combinação de W-CDMA e TDMA. Ele tenta atingir as metas do IMT-2000 adicionando multiplexação TDMA ao W-CDMA.

IMT-SC Esse padrão usa TDMA apenas.

IMT-FT Esse padrão usa uma combinação de FDMA e TDMA.

Quarta geração (4G)

Espera-se que a quarta geração da telefonia celular seja uma completa revolução no universo das comunicações sem fio. Alguns dos objetivos definidos pelo grupo de trabalho 4G são os seguintes:

a. Um sistema espectralmente eficiente.
b. Alta capacidade da rede.
c. Taxa de transferência de dados de 100 Mbit/s para acessos provenientes de um automóvel em movimento e de 1 Gbit/s para usuários estacionários.
d. Taxa de transferência de dados de pelo menos 100 Mbit/s entre quaisquer dois pontos no mundo.
e. *Handoff* fluído entre redes heterogêneas.
f. Conectividade completa e *roaming* global ao longo de múltiplas redes.
g. Alta qualidade de serviços para suporte a multimídia de próxima geração (o tema qualidade de serviço será discutido no Capítulo 8).
h. Interoperabilidade com padrões sem fio existentes.
i. Rede de comutação de pacotes inteiramente baseada em IP.

A quarta geração é totalmente baseada em comutação de pacotes (ao contrário do 3G) e suporta IPv6. Isto leva a uma melhor capacidade de prover *multicast*, segurança e otimização de rotas.

Esquema de acesso

Para aumentar a eficiência, a capacidade e a escalabilidade, novas técnicas de acesso estão sendo consideradas para o 4G. Por exemplo, o **FDMA Ortogonal** (OFDMA – Orthogonal FDMA) e o **FDMA Intercalado** (IFDMA – Interleaved FDMA) estão sendo considerados, respectivamente, para a recepção e envio de dados da próxima geração do **Sistema de Telecomunicações Móveis Universal** (UMTS – Universal Mobile Telecommunications Systemu). De forma similar, o **CDMA Multiportadora** (MC-CDMA – Multi-Carrier Code Division Multiple Access) está sendo proposto para o padrão IEEE 802.20.

Modulação

Um esquema mais eficiente de Modulação por Amplitude em Quadratura (o 64-QAM) está sendo proposto para uso nos padrões Evolução de Longo Prazo (LTE – Long Term Evolution).

Sistema de rádio

A quarta geração utiliza um sistema de **Rádio Definido por *Software*** (SDR – Software Defined Radio). Ao contrário de um rádio comum, que utiliza *hardware*, os componentes de um SDR são peças de *software* e, portanto, mais flexíveis. Um SDR pode alterar seu programa para deslocar suas frequências com o objetivo de amenizar interferências por sinais em frequências presentes no meio.

Antena

O sistema de antenas **Múltiplas Entradas e Múltiplas Saídas** (MIMO – Multiple-Input Multiple--Output) e **MU-MIMO (MIMO Multiusuário)**, uma vertente da área de antenas inteligentes, foi proposto para uso no 4G. Usando esse sistema de antenas em conjunto com multiplexação especial, o 4G permite que fluxos independentes sejam transmitidos simultaneamente a partir de todas as antenas, aumentando muitas vezes a taxa de transferência de dados. O MIMO também permite que o emissor e o receptor movam-se de forma coordenada para uma frequência livre quando houver interferência.

Aplicações

Com as taxas de transferências atuais de 15-30 Mbit/s, o 4G é capaz de fornecer aos usuários transmissões de TV de alta definição. Com taxas de 100 Mbit/s, o conteúdo de um DVD-5 pode ser obtido em cerca de cinco minutos para que possa ser assistido posteriormente.

6.2.3 Redes de satélites

Uma *rede de satélites* consiste em uma combinação de nós, alguns dos quais sendo satélites, que permitem comunicações de um ponto a outro do planeta Terra. Um nó da rede pode ser um satélite, uma estação terrestre ou um terminal ou telefone de usuário final. Embora um satélite natural, como a lua, possa ser usado como um nó na rede, a utilização de satélites artificiais é preferível porque podemos instalar equipamentos eletrônicos no satélite para regenerar sinais que tenham perdido parte de sua energia durante o trajeto. Outra restrição com relação ao uso de satélites naturais refere-se à distância deles até a Terra, o que leva a grandes atrasos na comunicação.

Redes de satélites são como redes de telefonia celular, no sentido que eles dividem o planeta em células. Os satélites podem fornecer capacidade de transmissão de e para qualquer local da Terra, não importando a distância. Essa vantagem permite que comunicações de alta qualidade sejam disponibilizadas em partes subdesenvolvidas do mundo sem a necessidade de investimentos vultosos em infraestrutura terrestre.

Órbitas

Um satélite artificial precisa ter uma *órbita*, o trajeto que ele segue ao redor da Terra. A órbita pode ser equatorial, inclinada ou polar, conforme mostra a Figura 6.46.

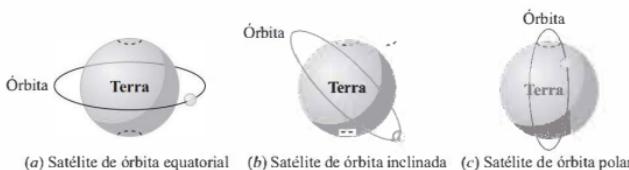



Figura 6.46 Órbitas de satélites.

O período de um satélite, que corresponde ao tempo necessário para que ele faça uma volta completa ao redor da Terra, é determinado pela lei de Kepler. Tal lei determina o período como uma função da distância do satélite ao centro da Terra.

Exemplo 6.4

Qual é o período da Lua, de acordo com a lei de Kepler?

$$\text{Período} = C \times \text{distância}^{1,5}$$

Nesta equação, C é uma constante que vale aproximadamente 1/100. O período é dado em segundos e a distância é dada em quilômetros.

Solução

A Lua está localizada a aproximadamente 384.000 km acima da Terra. O raio da Terra é de 6.378 km. Aplicando a fórmula, obtemos o seguinte.

$$\text{Período} = (1/100) \times (384.000 + 6.378)^{1,5} = 2.439.090 \text{ s} = 1 \text{ mês}$$

Exemplo 6.5

De acordo com a lei de Kepler, qual é o período de um satélite que esteja localizado em uma órbita a cerca de 35.786 km acima da Terra?

Solução

Aplicando a fórmula, obtemos o seguinte.

$$\text{Período} = (1/100) \times (35.786 + 6.378)^{1,5} = 86.579 \text{ s} = 24 \text{ h}$$

Isto significa que um satélite localizado a 35.786 km tem um período de 24 h, que é igual ao período de rotação da Terra. Diz-se que esse tipo de satélite é *estacionário* em relação à Terra. A órbita, conforme veremos, é denominada *órbita geoestacionária*.

Cobertura

Os satélites processam micro-ondas com antenas bidirecionais (linha de visada). Portanto, o sinal de um satélite é normalmente destinado a uma área específica denominada **área de cobertura**. No centro da área de cobertura, a potência do sinal é máxima. A potência diminui à medida que nos afastamos do centro da área de cobertura. A fronteira da área de cobertura corresponde ao local onde o nível de potência tem um limite predefinido.

Três categorias de satélites

Com base na localização da órbita, os satélites podem ser divididos em três categorias: **Órbita Terrestre Geoestacionária** (GEO – Geostationary Earth Orbit), **Órbita Terrestre Baixa** (LEO – Low-Earth-Orbit) e **Órbita Terrestre Média** (MEO – Medium-Earth-Orbit).

A Figura 6.47 mostra as altitudes dos satélites com relação à superfície da Terra. Existe apenas uma órbita, a uma altitude de 35.786 km, para os satélites GEO. Satélites MEO ficam localizados em altitudes entre 5.000 e 15.000 km. Satélites LEO normalmente encontram-se em uma altitude inferior a 2.000 km.

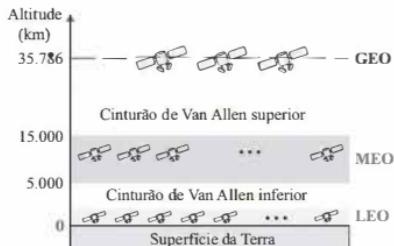



Figura 6.47 Altitudes das órbitas de satélites.

Uma razão para haver diferentes órbitas é a existência de dois cinturões de Van Allen. Um cinturão de Van Allen é uma camada que contém partículas carregadas. Um satélite orbitando um desses dois cinturões seria totalmente destruído pelas partículas carregadas de energia. As órbitas MEO ficam localizadas entre os dois cinturões.

Bandas de frequência para comunicação via satélite

As frequências de micro-ondas reservadas para a comunicação via satélite encontram-se na faixa dos giga-hertz (GHz). Cada satélite envia e recebe dados usando duas bandas distintas. A transmissão da Terra para o satélite é chamada *uplink* ou *ligação ascendente*. A transmissão do satélite para a Terra é chamada *downlink* ou *ligação descendente*. A Tabela 6.5 mostra os nomes das bandas e as frequências para cada faixa.

Tabela 6.5 Bandas de frequência usadas por satélites.

Banda	*Downlink*, GHz	*Uplink*, GHz	Largura de banda, MHz
L	1,5	1,6	15
S	1,9	2,2	70
C	4,0	6,0	500
Ku	11,0	14,0	500
Ka	20,0	30,0	3500

Satélites GEO

A propagação em linha de visada requer que as antenas emissora e receptora estejam voltadas uma para a outra o tempo todo (uma antena deve ser capaz de avistar a outra). Por isso, um satélite que se move mais rapidamente ou mais lentamente do que a rotação da Terra é útil somente por curtos períodos. Para assegurar uma comunicação constante, o satélite deve mover-se à mesma velocidade que a Terra, de modo que ele aparente estar fixo acima de um determinado ponto. Esses satélites são chamados *geoestacionários*.

Como a velocidade orbital baseia-se na distância do satélite ao planeta, só pode haver uma órbita geoestacionária, que se encontra no plano equatorial e fica a aproximadamente 35.786 quilômetros da superfície da Terra.

Porém, um único satélite geoestacionário não é capaz de cobrir a Terra toda. Um satélite em órbita é capaz de manter contato em linha de visada com um grande número de estações, mas a curvatura da Terra ainda impede que uma grande parte do planeta seja avistada. São necessários no mínimo três satélites equidistantes uns dos outros em órbita terrestre geoestacionária (GEO) para que a transmissão tenha cobertura global completa. A Figura 6.48 mostra três satélites em órbita geoestacionária ao redor do equador, cada um deles separado do outro por 120°. A perspectiva da figura é a do Polo Norte.

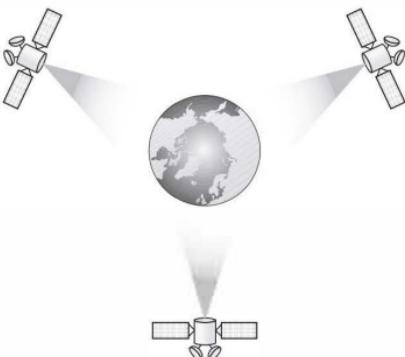



Figura 6.48 Satélites em órbita geoestacionária.

Satélites MEO

Satélites MEO (*Medium-Earth-Orbit*, ou Órbita Terrestre Média) são aqueles posicionados entre os dois cinturões de Van Allen. Um satélite nessa órbita demora cerca de 6 a 8 horas para circundar a Terra.

Sistema de Posicionamento Global

Um exemplo de um sistema de satélites MEO é o **Sistema de Posicionamento Global** (GPS – Global Positioning System), criado e operado pelo Departamento de Defesa dos Estados Unidos, que orbita a uma altitude de cerca de 18.000 km (11.000 milhas) acima da Terra. O sistema é constituído por 24 satélites e é usado para navegação por terra, mar e ar, fornecendo hora e localização para veículos e embarcações. O GPS utiliza 24 satélites em seis órbitas, conforme mostra a Figura 6.49. As órbitas e as localizações dos satélites em cada órbita foram projetadas de tal forma que, a qualquer momento, quatro satélites sejam visíveis a partir de qualquer ponto na Terra. Um receptor de GPS tem um sistema interno que informa a posição atual de cada satélite.

Trilateração O GPS opera com base em um princípio denominado **trilateração**. Os termos *trilateração* e *triangulação* são normalmente usados como sinônimos. Usamos a palavra *trilateração*, que significa o uso de três distâncias, em vez de **triangulação**, que pode ser interpretado como o uso de três ângulos. Em um avião, se soubermos nossa distância com relação a três pontos, sabemos exatamente onde estamos. Digamos que estamos a 10 km de distância do ponto A, a 12 km de distância do ponto B e a 15 km de distância do ponto C. Se desenharmos três círculos cujos centros sejam A, B e C, devemos estar em algum lugar no círculo A, em algum lugar no

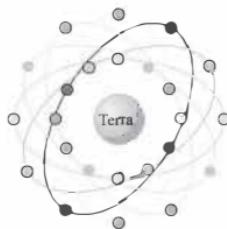



Figura 6.49 Órbitas dos satélites do Sistema de Posicionamento Global (GPS).

círculo B, e em algum lugar no círculo C. Esses três círculos se interceptam em um único ponto (se os valores de nossas distâncias estiverem corretos); essa é nossa posição. A Figura 6.50*a* ilustra o conceito.

No espaço tridimensional, a situação é diferente. Três esferas se interceptam em dois pontos, conforme mostra a Figura 6.50*b*. Precisamos de pelo menos quatro esferas para determinar nossa posição exata no espaço (longitude, latitude e altitude). Entretanto, se conhecermos fatos adicionais sobre nossa localização (por exemplo, sabemos que não estamos dentro do oceano ou em algum lugar no espaço), três esferas são suficientes, pois um dos dois pontos no qual as esferas se interceptam é tão improvável que o outro pode ser selecionado sem dúvidas.

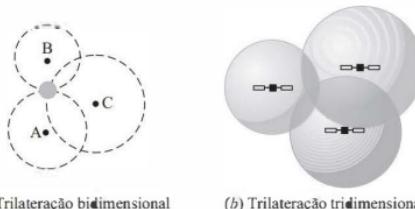



Figura 6.50 Trilateração em um avião.

Medindo a distância O princípio da triangulação permite determinar nossa localização na Terra se soubermos a que distância nos encontramos de três satélites e a posição de cada satélite, que pode ser calculada por um receptor GPS (usando a trajetória predeterminada dos satélites). O receptor GPS precisa, então, determinar a sua distância com relação a pelo menos três satélites de GPS (centros das esferas). A medição da distância é realizada usando um princípio conhecido como *one-way ranging*, ou *medição unidirecional*. Por ora, consideremos que todos os satélites de GPS e o receptor terrestre estejam sincronizados. Cada um dos 24 satélites transmite sincronamente um sinal complexo, sendo que o sinal de cada satélite apresenta um padrão único. O computador no receptor mede o atraso entre os sinais provenientes dos satélites e sua cópia dos sinais originais para determinar as distâncias até os satélites.

Sincronização A discussão anterior se baseou na suposição de que os relógios dos satélites estão sincronizados entre eles e também com o relógio do receptor. Satélites usam relógios atômicos, que são precisos e podem funcionar de forma síncrona uns com os outros. O relógio do receptor, no

entanto, é um relógio normal de quartzo (um relógio atômico custa mais de US$ 50.000), e não existe uma forma de sincronizá-lo com os relógios dos satélites. Há uma imprecisão desconhecida entre os relógios dos satélites e o relógio do receptor, o que introduz uma imprecisão correspondente no cálculo da distância. Devido a essa imprecisão, a distância medida é denominada *pseudodistância*.

O GPS utiliza uma solução elegante para o problema da imprecisão do relógio, admitindo que o valor da imprecisão é igual para todos os satélites que estão sendo usados. O cálculo da posição se resume a determinar quatro incógnitas: as coordenadas x_r, y_r e z_r do receptor, e a imprecisão em comum do relógio, dt. Para determinar esses quatro valores desconhecidos, precisamos de pelo menos quatro equações. Isto significa que precisamos medir pseudodistâncias com relação a quatro satélites em vez de três. Se denotarmos as quatro pseudodistâncias medidas por PR_1, PR_2, PR_3 e PR4, e denotarmos as coordenadas de cada satélite por x_i, y_{ii} e z_i (para $i = 1$ até 4), podemos determinar os quatro valores desconhecidos anteriormente mencionados usando as seguintes quatro equações (os valores desconhecidos são mostrados em cinza).

$$PR_1 = [(x_1 - x_r)^2 + (y_1 - y_r)^2 + (z_1 - z_r)^2]^{1/2} + c \times dt$$
$$PR_2 = [(x_2 - x_r)^2 + (y_2 - y_r)^2 + (z_2 - z_r)^2]^{1/2} + c \times dt$$
$$PR_3 = [(x_3 - x_r)^2 + (y_3 - y_r)^2 + (z_3 - z_r)^2]^{1/2} + c \times dt$$
$$PR_4 = [(x_4 - x_r)^2 + (y_4 - y_r)^2 + (z_4 - z_r)^2]^{1/2} + c \times dt$$

As coordenadas usadas nas fórmulas anteriores são dadas em um espaço de referência ECEF (*Earth-Centered Earth-Fixed*, ou Centrado e Fixo na Terra), o que significa que a origem do espaço de coordenadas é o centro da Terra e que o espaço de coordenadas gira com a Terra. Isto implica que as coordenadas ECEF de um ponto fixo sobre a superfície da Terra não mudam.

Aplicação O GPS é usado por forças militares. Por exemplo, milhares de receptores GPS portáteis foram usados durante a Guerra do Golfo Pérsico por soldados, veículos e helicópteros. Outro uso do GPS é na navegação. O condutor de um automóvel pode determinar a localização de seu veículo. O condutor pode, em seguida, consultar uma base de dados na memória do automóvel* e ser guiado até seu destino. Em outras palavras, o GPS indica a localização do carro e a base de dados utiliza essa informação para encontrar um caminho até o destino. Outra aplicação muito interessante é a sincronização de relógios. Conforme mencionado anteriormente, o sistema de telefonia celular IS-95 usa o GPS para a sincronização temporal entre as estações-base.

Satélites LEO

Satélites LEO (*Low-Earth-Orbit*, ou Órbita Terrestre Baixa) têm órbitas polares. Sua altitude fica entre 500 e 2.000 km, com um período de rotação de 90 a 120 minutos. O satélite tem uma velocidade de 20.000 a 25.000 km/h. Um sistema LEO geralmente apresenta uma cobertura do tipo celular, semelhante ao sistema de telefonia celular. A área de cobertura normalmente tem um diâmetro de 8.000 km. Como os satélites LEO ficam próximos à Terra, o atraso de propagação para a ida e volta do sinal é normalmente menor que 20 ms, valor aceitável para comunicações de áudio.

Um sistema LEO é composto por uma constelação de satélites que operam em conjunto como uma rede; cada satélite atua como um comutador. Satélites que estejam próximos um do outro são conectados por meio de Enlaces Intersatélite (ISLs – Intersatellite Links). Um sistema móvel se comunica com o satélite por meio de um Enlace Móvel de Usuário (UML – User Mobile Link). Um satélite pode também se comunicar com uma estação terrestre (*gateway*) por meio de um Enlace de *Gateway* (GWL – Gateway Link). A Figura 6.51 mostra uma rede de satélites LEO comum.

Os satélites LEO podem ser divididos em três categorias: *little LEO*, *big LEO* e *broadband LEO*. Os satélites *little* LEO (ou LEO pequenos) operam em uma frequência inferior a 1 GHz. Eles são usados principalmente para mensagens que exigem uma baixa taxa de transferência de dados.

* N. de T.: A base de memória do GPS costuma incluir informações como mapas das vias, condições de tráfego, dentre outras.

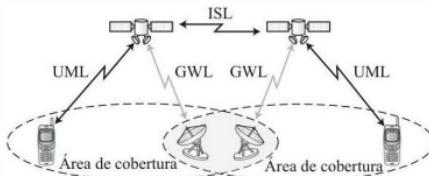



Figura 6.51 Sistema de satélites LEO.

Os satélites *big* LEO (ou LEO grandes) operam em frequências entre 1 e 3 GHz. O sistema **Globalstar** é um exemplo de sistema de satélites *big* LEO. Ele usa 48 satélites em seis órbitas polares, sendo que cada órbita hospeda oito satélites. As órbitas ficam a uma altitude de quase 1.400 km. O sistema **Iridium** também é um exemplo que usa satélites big LEO. O sistema Iridium tem 66 satélites distribuídos em seis órbitas, com 11 satélites em cada uma. As órbitas ficam a uma altitude de 750 km. Os satélites em cada órbita ficam separados um do outro por aproximadamente 32° de latitude. Os satélites do tipo *broadband* LEO (ou LEO de banda larga) fornecem uma capacidade de comunicação semelhante àquela obtida com redes de fibra óptica. O primeiro sistema *broadband* LEO foi o **Teledesic**, um é um sistema de satélites que fornece comunicação similar à da fibra óptica (canais de banda larga, reduzida taxa de erros e baixo atraso). Seu principal objetivo é fornecer acesso de banda larga à Internet para usuários do mundo todo. Ele é também conhecido como "Internet no céu." O projeto desse sistema foi iniciado em 1990 por Craig McCaw e Bill Gates; mais tarde, outros investidores juntaram-se ao consórcio. O projeto está programado para estar totalmente funcional em um futuro próximo.

6.3 IP MÓVEL

Como os computadores móveis e pessoais como os *notebooks* vêm se tornando cada vez mais populares, é necessário considerarmos o desenvolvimento de sistemas com suporte a IP móvel. Essa é a extensão do protocolo IP que permite que computadores móveis se conectem à Internet de qualquer lugar onde tal conexão seja possível. Nesta seção, discutimos essa questão.

6.3.1 Endereçamento

O principal problema que deve ser resolvido quando se deseja fornecer comunicação móvel usando o protocolo IP refere-se ao endereçamento.

Estações fixas

O endereçamento IP original baseia-se no pressuposto de que as estações são fixas, ficando ligadas a uma rede específica. Os roteadores usam o endereço IP para rotear um datagrama IP. Conforme vimos no Capítulo 5, um endereço IP tem duas partes: um prefixo e um sufixo. O prefixo associa uma estação a uma rede. Por exemplo, o endereço IP 10.3.4.24/8 define uma estação ligada à rede 10.0.0.0/8. Isto implica que uma estação na Internet não tenha um endereço de rede que ela possa carregar consigo de um lugar para outro. O endereço é válido somente enquanto a máquina está conectada àquela rede. Se a estação trocar de rede, o endereço não é mais válido. Os roteadores usam essa associação para encaminhar pacotes; eles usam o prefixo para entregar o pacote à rede à qual a estação está conectada. Esse esquema funciona perfeitamente com **estações fixas**.

Os endereços IP foram projetados para funcionar com estações fixas, pois parte do endereço define a rede à qual a estação está conectada.

Estações móveis

Quando consideramos estações que se movem de uma rede para outra, a estrutura do endereçamento IP precisa ser modificada. Diversas soluções foram propostas.

Alteração do endereço

Uma solução simples é permitir que a **estação móvel** troque seu endereço quando for para a nova rede. A estação pode usar o DHCP (ver Capítulo 4) para obter um novo endereço que associe a estação à nova rede. Essa abordagem tem vários inconvenientes. Em primeiro lugar, os arquivos de configuração precisariam ser alterados. Em segundo lugar, cada vez que o computador se movesse de uma rede para outra, sua placa de rede teria que ser reinicializada. Em terceiro lugar, as tabelas DNS (ver Capítulo 2) precisariam ser atualizadas, de modo que todas as outras estações na Internet ficassem cientes da mudança. Em quarto lugar, se a estação se movesse de uma rede para outra durante uma transmissão, a transferência dos dados seria interrompida. Isto acontece porque as portas e os endereços IP do cliente e do servidor devem permanecer inalterados durante toda a conexão.

Dois endereços

A abordagem mais viável consiste em utilizar dois endereços. A estação tem seu endereço original, conhecido como **endereço nativo** (*home address*), e um endereço temporário, denominado **endereço de tratamento** (*care-of address**). O endereço nativo não se altera; ele associa a estação à sua **rede nativa**, também conhecida como **rede local** (*home network*), a rede que é a residência permanente da estação. O endereço de tratamento é temporário. Quando uma estação se move de uma rede para outra, o endereço de tratamento muda; ele está associado à **rede externa**, também denominada **rede visitada** (*foreign network*), que corresponde à rede para a qual a estação se moveu. A Figura 6.52 ilustra o conceito.

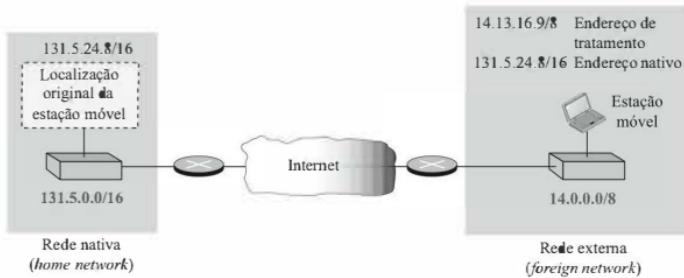



Figura 6.52 Endereço nativo (*home address*) e endereço de tratamento (*care-of address*).

* N. de T.: O nome *care-of address* tem sua origem no sistema de correios tradicional, no qual se usa a expressão *care-of* ("aos cuidados de"), para indicar o nome de um destinatário específico da carta (possivelmente diferente do proprietário da residência).

O IP móvel emprega dois endereços para uma estação móvel: um endereço nativo (*home address*) e um endereço de tratamento (*care-of address*). O endereço nativo não se altera; o endereço de tratamento muda à medida que a estação móvel se move de uma rede para outra.

Quando uma estação móvel visita uma rede externa, aquela estação recebe seu endereço de tratamento durante o processo de descoberta de agente e registro, descrito mais adiante.

6.3.2 Agentes

Para fazer com que a mudança do endereço seja transparente para o restante da Internet, são necessários um **agente nativo**, também conhecido como *home agent*, e um **agente externo**, também denominado *foreign agent*. A Figura 6.53 mostra a posição de um agente nativo com relação à rede nativa e de um agente externo com relação à rede externa.

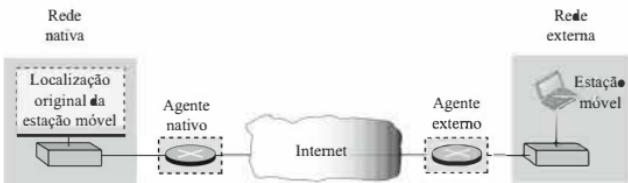



Figura 6.53 Agente nativo (*home agent*) e agente externo (*foreign agent*).

Mostramos os agentes nativos e externos como roteadores, mas precisamos enfatizar que sua função específica como agentes é realizada na camada de aplicação. Em outras palavras, ambos atuam como roteadores e como aplicativos.

Agente nativo

O agente nativo é geralmente um roteador conectado à rede original da estação móvel. O agente nativo atua em nome da estação móvel quando uma estação remota envia um pacote para a móvel. O agente nativo recebe o pacote e o envia para o agente externo.

Agente externo

O agente externo é geralmente um roteador conectado à rede visitada. Ele recebe os pacotes enviados pelo agente local e os entrega para a estação móvel.

A estação móvel também pode atuar como um agente externo. Em outras palavras, a estação móvel e o agente externo podem ser a mesma entidade. Entretanto, para fazer isso, a estação móvel deve, ela mesma, ser capaz de receber um endereço de tratamento, o que pode ser feito usando o DHCP. Além disso, a estação móvel precisa do *software* necessário para se comunicar com o agente nativo e ter dois endereços: o seu endereço nativo e o de tratamento. Essa abordagem de endereçamento duplo deve ser transparente para os programas da camada de aplicação.

Quando a estação móvel atua como um agente externo, o endereço de tratamento é denominado um **endereço de tratamento coalocado** (*collocated care-of address*).

Quando a estação móvel e o agente externo são a mesma entidade, o endereço de tratamento é denominado um endereço de tratamento coalocado (*collocated care-of address*).

A vantagem de usar um endereço de tratamento coalocado é que a estação móvel pode mover-se para qualquer rede sem se preocupar com a disponibilidade de um agente externo. A desvantagem é que a estação móvel precisa de um *software* extra para poder agir como seu próprio agente externo.

6.3.3 Três fases

Para se comunicar com uma estação remota, uma estação móvel passa por um processo em três fases: descoberta de agente, registro e transferência de dados, conforme mostra a Figura 6.54.

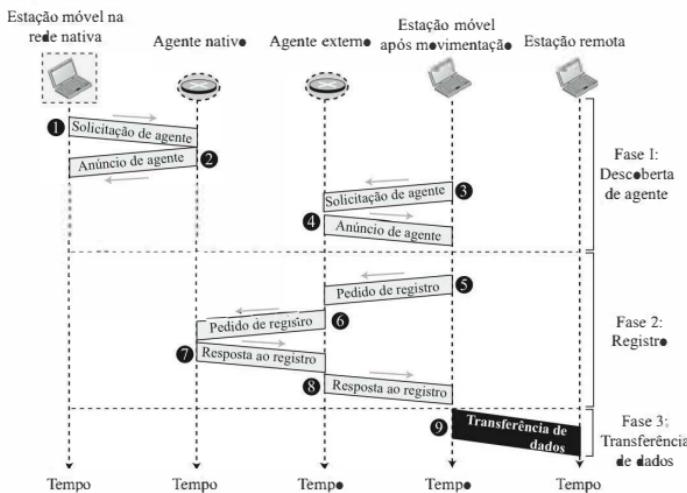



Figura 6.54 Estação remota e comunicação com estação móvel.

A primeira fase, de descoberta de agente, envolve a estação móvel, o agente externo e o agente nativo. A segunda fase, o registro, também envolve a estação móvel e os dois agentes. Finalmente, na terceira fase, a estação remota também está envolvida. Cada fase é discutida separadamente.

Descoberta de agente

A primeira fase na comunicação móvel, a *descoberta de agente*, é composta por duas subfases. Uma estação móvel deve descobrir (determinar o endereço de) um agente nativo antes de deixar a sua rede nativa. Ele precisa também descobrir um agente externo depois de ter se movido para uma rede externa. Essa descoberta consiste em determinar o endereço de tratamento, bem como o endereço do agente externo. A descoberta envolve dois tipos de mensagens: anúncio e solicitação.

Anúncio de agente

Quando um roteador anuncia a sua presença em uma rede usando uma mensagem ICMP de anúncio de roteador, ele pode concatenar um ***anúncio de agente*** ao pacote caso atue como um agente. A Figura 6.55 mostra como um anúncio do agente é enviado juntamente com o pacote de anúncio de roteador.

Mensagem de anúncio ICMP			
Tipo	Comprimento	Número de sequência	
Tempo de vida		Código	Reservado
Endereço de tratamento (apenas no caso de agentes externos)			

Figura 6.55 Anúncio de agente.

O IP móvel não usa um novo tipo de pacote para o anúncio de agentes, mas, sim, o pacote ICMP de anúncio de roteador, concatenando uma mensagem de anúncio de agente.

As descrições dos campos são as seguintes:

- **Tipo.** O campo de tipo, de 8 *bits*, tem o valor fixo 16.
- **Comprimento.** O campo de comprimento, de 8 *bits*, especifica o comprimento total da mensagem de extensão (não o comprimento da mensagem de anúncio ICMP).
- **Número de sequência.** O campo de número de sequência, de 16 *bits*, contém o número da mensagem. O destinatário pode usar o número de sequência para determinar se alguma mensagem foi perdida.
- **Tempo de vida.** O campo de tempo de vida especifica o número de segundos durante o qual agente aceitará solicitações. Se o valor for uma sequência de *bits* 1, o tempo de vida é infinito.
- **Código.** O campo de código é um campo de 8 *bits* contendo indicadores, sendo que cada *bit* assume o valor ativo (1) ou inativo (0). Os significados dos *bits* são mostrados na Tabela 6.6.

Tabela 6.6 *Bits* de código.

Bit	Significado
0	Registro necessário. Endereços de tratamento coalocados não são permitidos.
1	O agente está ocupado e não está aceitando registros neste momento.
2	O agente atua como um agente nativo.
3	O agente atua como um agente externo.
4	O agente usa encapsulamento mínimo.
5	O agente usa Encapsulamento de Roteamento Genérico (GRE – Generic Routing Encapsulation).
6	O agente suporta compressão de cabeçalho.
7	Não utilizado (0).

- **Endereços de tratamento.** Este campo contém uma lista de endereços disponíveis para uso como endereços de tratamento. A estação móvel pode escolher um deles. A seleção desse endereço de tratamento é anunciada no pedido de registro. Perceba que esse campo é usado apenas por agentes externos.

Solicitação de agente

Quando uma estação móvel se move para uma nova rede, mas não recebe anúncios de agente, ela pode iniciar uma *solicitação de agente* e pode usar a mensagem ICMP de solicitação de roteador para informar um agente de que precisa de ajuda.

O IP móvel não usa um novo tipo de pacote para a solicitação de agente; mas, sim, o pacote ICMP de solicitação de roteador.

Registro

A segunda fase na comunicação móvel é o *registro*. Depois que a estação móvel se moveu para uma rede externa e descobriu o agente externo, ela deve se registrar. Existem quatro aspectos no registro:

1. A estação móvel deve se registrar junto ao agente externo.
2. A estação móvel deve se registrar junto ao seu agente nativo. Isto normalmente é feito pelo agente externo em nome da estação móvel.
3. A estação móvel deve renovar o registro se ele expirar.
4. A estação móvel deve cancelar seu registro quando ela voltar para sua rede nativa.

Pedido e resposta

Para efetuar o registro junto ao agente externo e ao agente nativo, a estação móvel usa mensagens de pedido de registro e de resposta ao registro, conforme mostra a Figura 6.54.

Pedido de registro Um pedido de registro é enviado por uma estação móvel ao agente externo para registrar seu endereço de tratamento e também para anunciar seu endereço nativo e o endereço do agente nativo. O agente externo, após receber e registrar o pedido, encaminha a mensagem para o agente nativo. Perceba que este agora sabe o endereço do agente externo porque o pacote IP, envolvido no encaminhamento, carrega o endereço IP do agente externo como endereço de origem. A Figura 6.56 mostra o formato do pedido de registro.

Tipo	Indicadores	Tempo de vida
Endereço nativo		
Endereço do agente nativo		
Endereço de tratamento		
Identificação		
Extensões ...		

Figura 6.56 Formato da mensagem de pedido de registro.

As descrições dos campos são apresentadas a seguir:

- **Tipo.** O campo de tipo, de 8 *bits*, especifica o tipo da mensagem. Para uma mensagem de pedido, o valor desse campo é 1.

- **Marcadores (*flags*).** O campo de indicadores, de 8 *bits*, especifica as informações de encaminhamento. O valor de cada *bit* pode ser 0 ou 1. O significado de cada *bit* é mostrado na Tabela 6.7.

Tabela 6.7 *Bits* do campo de indicadores na mensagem de pedido de registro.

Bit	Significado
0	Estação móvel pede que o agente nativo conserve seu antigo endereço de tratamento.
1	Estação móvel pede que o agente nativo lhe repasse qualquer mensagem de *broadcast*.
2	Estação móvel está usando um endereço de tratamento coalocado.
3	Estação móvel pede que o agente nativo use encapsulamento mínimo.
4	Estação móvel pede que seja usado Encapsulamento de Roteamento Genérico (GRE).
5	Estação móvel pede que seja realizada compressão de cabeçalho.
6-7	*Bits* reservados.

- **Tempo de vida.** Campo que especifica o número de segundos durante o qual o registro é válido. Se o campo for uma sequência de 0s, a mensagem de pedido está requisitando um cancelamento de registro. Se for uma sequência de 1s, o tempo de vida é infinito.
- **Endereço nativo.** Campo que contém o endereço permanente da estação móvel (seu primeiro endereço).
- **Endereço do agente nativo.** Campo que contém o endereço do agente nativo.
- **Endereço de tratamento.** Campo que corresponde ao endereço temporário da estação móvel (seu segundo endereço).
- **Identificação.** Campo que contém um número de 64 *bits* inserido pela estação móvel no pedido e repetido na mensagem de resposta. Ele identifica a resposta correspondente a um pedido.
- **Extensões.** As extensões de comprimento variável são usadas para autenticação. Elas permitem que um agente nativo autentique o agente móvel. Discutimos autenticação no Capítulo 10.

Resposta ao registro Uma resposta as registro é enviada pelo agente nativo para o agente externo e, então, encaminhada à estação móvel. A resposta confirma a aceitação ou informa a recusa do pedido de registro. A Figura 6.57 mostra o formato da mensagem de resposta ao registro.

Tipo	Código	Tempo de vida
Endereço nativo		
Endereço do agente nativo		
Identificação		
Extensões ...		

Figura 6.57 Formato da mensagem de resposta ao registro.

Os campos são semelhantes àqueles do pedido de registro, com as seguintes exceções. O valor do campo de tipo é 3. O campo de código substitui o campo de indicadores e contém o resultado do pedido de registro (aceitação ou recusa). O endereço de tratamento não precisa ser incluído na mensagem.

Encapsulamento

Mensagens de registro são encapsuladas em um datagrama de usuário UDP. Agentes usam a porta bem conhecida 434; estações móveis usam uma porta efêmera.

Pedidos de registro e respostas ao registro são enviados via UDP usando a porta bem conhecida 434.

Transferência de dados

Após as fases de descoberta de agente e de registro, uma estação móvel pode se comunicar com uma estação remota. A Figura 6.58 ilustra a ideia.

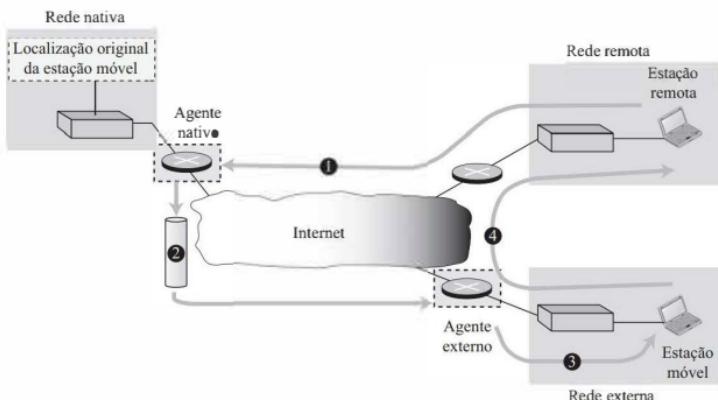



Figura 6.58 Transferência de dados.

Da estação remota até o agente nativo

Quando uma estação remota quer enviar um pacote para a estação móvel, a remota usa seu próprio endereço como o endereço de origem e o endereço nativo da estação móvel como o de destino. Em outras palavras, a estação remota envia um pacote como se a estação móvel estivesse em sua rede nativa. Ele, no entanto, é interceptado pelo agente nativo, que age como se fosse a estação móvel. Isto é feito usando uma técnica conhecida como *proxy* ARP*. O caminho 1 na Figura 6.58 ilustra essa etapa.

* N. de T.: Na técnica de *proxy* ARP, um dispositivo em uma rede responde a um pedido ARP para um endereço de rede que não esteja naquela rede. Esse dispositivo conhece o local para onde o tráfego deve ser encaminhado, e oferece seu próprio endereço MAC na resposta, como se dissesse "me envie os dados e eu os encaminharei para o destino correto".

Do agente nativo até o agente externo

Após receber o pacote, o agente nativo o envia para o agente externo, usando o conceito de tunelamento discutido no Capítulo 4. O agente nativo encapsula o pacote IP inteiro dentro de outro pacote IP usando seu próprio endereço como endereço de origem e o endereço do agente externo como endereço de destino. O caminho 2 na Figura 6.58 ilustra esse passo.

Do agente externo até a estação móvel

Quando o agente externo recebe o pacote, ele recupera o pacote original. Entretanto, como o endereço de destino dele corresponde ao endereço nativo da estação móvel, o agente externo consulta uma tabela de registro para determinar o endereço de tratamento da estação móvel. Se isto não fosse feito, o pacote seria simplesmente enviado de volta para a rede nativa. O pacote é, então, enviado para o endereço de tratamento. O caminho 3 na Figura 6.58 ilustra esse passo.

Da estação móvel até a estação remota

Quando uma estação móvel deseja enviar um pacote para uma estação remota (por exemplo, uma resposta para o pacote que ela recebeu), ela envia o pacote normalmente. A estação móvel prepara um pacote com seu endereço nativo como endereço de origem e com o endereço da estação remota como o endereço de destino. Embora o pacote venha da rede externa, ele apresenta o endereço nativo da estação móvel. O caminho 4 na Figura 6.58 ilustra esse passo.

Transparência

Nesse processo de transferência de dados, a estação remota não toma conhecimento de qualquer movimentação da estação móvel. A estação remota envia pacotes usando o endereço nativo da estação móvel como endereço de destino; ela recebe pacotes cujo endereço de origem é o endereço nativo da estação móvel. A movimentação é totalmente transparente. O restante da Internet não fica ciente da mobilidade de uma estação que esteja se movendo.

A movimentação da estação móvel é transparente para o restante da Internet.

6.3.4 Ineficiência no IP móvel

A comunicação envolvendo o IP móvel pode ser ineficiente. A ineficiência pode ser intensa ou moderada. O caso intenso é denominado *double crossing, travessia dupla* ou *2X*. O caso moderado é denominado *roteamento triangular*.

Travessia dupla

A **travessia dupla** ocorre quando uma estação remota se comunica com uma estação móvel que está visitando a rede (ou sub-rede) daquela estação remota (ver Figura 6.59).

Quando a estação móvel envia um pacote para a estação remota, não há ineficiência; a comunicação é local. Entretanto, quando a estação remota envia um pacote para a estação móvel, o pacote passa pela Internet duas vezes.

Como um computador normalmente se comunica com outros computadores locais (princípio da localidade), a ineficiência da travessia dupla é significativa.

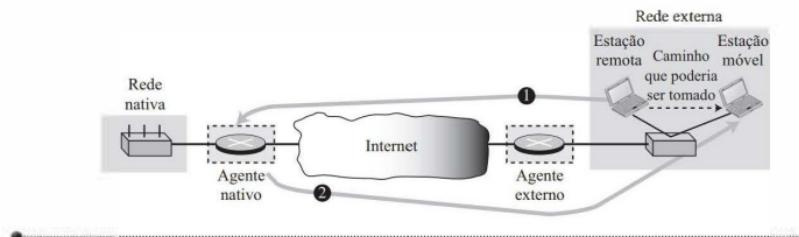



Figura 6.59 Travessia dupla.

Roteamento triangular

O **roteamento triangular,** o caso menos grave, ocorre quando a estação remota se comunica com uma estação móvel que não está conectada à mesma rede (ou sub-rede) que a estação remota. Quando a estação móvel envia um pacote para a estação remota, não há ineficiência. Entretanto, quando a estação remota envia um pacote para a estação móvel, o pacote vai da estação remota até o agente nativo e é, então, enviado para a estação móvel. O pacote viaja os dois lados de um triângulo, em vez de apenas um lado (ver Figura 6.60).

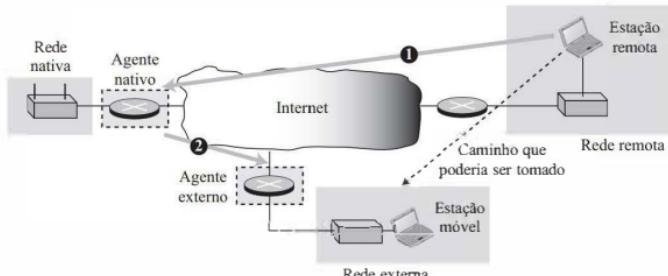



Figura 6.60 Roteamento triangular.

Solução

Uma solução para ineficiências é fazer a estação remota associar o endereço de tratamento ao endereço nativo de uma estação móvel. Por exemplo, quando um agente nativo recebe o primeiro pacote destinado a uma estação móvel, ele o encaminha para o agente externo; poderia também enviar um *pacote de atualização de associação* para a estação remota, para que futuros pacotes destinados à estação móvel possam ser enviados ao endereço de tratamento. A estação remota pode manter essas informações em uma unidade de armazenamento temporário.

O problema com essa estratégia é que a entrada armazenada pela estação remota pode ficar desatualizada quando a estação móvel mudar de rede. Nesse caso, o agente nativo precisa enviar um *pacote de aviso* para a estação remota com o objetivo de informá-la dessa mudança.

6.4 MATERIAL DO FINAL DO CAPÍTULO

6.4.1 Leitura adicional

Para mais detalhes sobre os assuntos discutidos neste capítulo, recomendamos os seguintes livros e RFCs. Os itens entre colchetes [...] referem-se à lista de referências no final do texto.

Livros

Diversos livros abordam questões discutidas neste capítulo, incluindo [Sch 03], [Gas 02]. [For 03], [Sta 04], [Sta 02], [Kei 02], [Jam 03], [AZ 03], [Tan 03], [Cou 01], [Com 06], [G & W 04] e [PD 03].

RFCs

Diversas RFCs discutem o IP móvel em particular: RFC 1701, RFC 2003, RFC 2004, RFC 3024, RFC 3344 e RFC 3775.

6.4.2 Termos-chave

- Acesso Múltiplo por Divisão de Código (CDMA – Code Division Multiple Access)
- Acesso Múltiplo por Divisão de Frequência (FDMA – Frequency-Division Multiple Access)
- Acesso Múltiplo por Divisão de Tempo (TDMA – Time-Division Multiple Access)
- agente externo (*foreign agent*)
- agente nativo (*home agent*)
- AMPS digital (D-AMPS)
- antena de Múltiplas Entradas e Múltiplas Saídas (MIMO – Multiple-Input Multiple-Output)
- antena MIMO multiusuário (MU-MIMO)
- área de cobertura
- Bluetooth
- canalização
- CDMA Multiportadora (MC-CDMA – Multi-Carrier CDMA)
- Central de Comutação Móvel (MSC – Mobile Switching Center)
- Chaveamento de Código Complementar (CCK – Complementary Code Keying)
- Comunicação para Internet Móvel 2000 (IMT-2000 – Internet Mobile Communication 2000)
- Conjunto Básico de Serviços (BSS – Basic Service Set)
- Conjunto Estendido de Serviços (ESS – Extended Service Set)
- endereço de tratamento (*care-of address*)
- endereço de tratamento coalocado (*collocated care-of address*)
- endereço nativo (*home address*)
- Enlace Assíncrono Sem Conexão (ACL – Asynchronous Connectionless Link)
- Enlace Síncrono Orientado a Conexão (SCO – Synchronous Connection-Oriented)
- Equipamento nas Instalações do Cliente (CPE – Customer Premises Equipment)
- Espaço Curto Entre Quadros (SIFS – Short Interframe Space)
- Espaço entre Quadros (IFS – Interframe Space)
- Espaço entre Quadros DCF (DIFS – DCF Interframe Space)
- Espalhamento Espectral por Saltos em Frequências (FHSS – Frequency Hopping Spread Spectrum)
- Espalhamento Espectral por Sequência Direta (DSSS – Direct Sequence Spread Spectrum)
- Espalhamento Espectral por Sequência Direta de Alta Taxa (HR-DSSS – High-Rate Direct-Sequence Spread Spectrum)
- estação fixa
- estação móvel
- fator de reuso
- FDMA Intercalado (IFDMA – Interleaved FDMA)
- FDMA ortogonal (OFDMA – Orthogonal FDMA)
- Função de Coordenação Distribuída (DCF – Distributed Coordination Function)
- Função de Coordenação Pontual (PCF – Point Coordination Function)
- Globalstar

- Interoperabilidade Mundial para Acesso por Micro-ondas (WiMAX – Worldwide Interoperability for Microwave Access)
- Iridium
- mobilidade sem transição
- Modulação por Posição de Pulso (PPM – Pulse Position Modulation)
- Multiplexação por Divisão de Frequências Ortogonais (OFDM – Orthogonal Frequency-Division Multiplexing)
- Órbita Terrestre Baixa (LEO – Low-Earth-Orbit)
- Órbita Terrestre Geoestacionária (GEO – Geostationary Earth Orbit)
- Órbita Terrestre Média (MEO – Medium-Earth-Orbit)
- Padrão Provisório 95 (IS-95 – Interim Standard 95)
- *piconet*
- Ponto de Acesso (AP – Access Point)
- Protocolo de Adaptação e Controle de Enlace Lógico (L2CAP – Logical Link Control and Adaptation Protocol)
- quadro de sinalização (*beacon frame*)
- Rádio Definido por *Software* (SDR – Software Defined Radio)
- rede *ad hoc*
- rede externa (*foreign network*)
- rede nativa (*home network*)
- rede pessoal (PAN – Personal Area Network)
- *roaming*
- roteamento triangular
- OFDMA escalável (SOFDMA – Scalable OFDMA)
- *scatternet*
- Sistema Avançado de Telefonia Móvel (AMPS – Advanced Mobile Phone System)
- Sistema de Antenas Adaptativas (AAS – Adaptive Antenna System)
- Sistema de Posicionamento Global (GPS – Global Positioning System)
- Sistema de Telecomunicações Móveis Universal (UMTS – Universal Mobile Telecommunications System)
- Sistema Global para Comunicações Móveis (GSM – Global System for Mobile Communication)
- Tabela de Walsh
- TDMA com Duplexação por Divisão de Tempo (TDD-TDMA – Time-Division Duplex TDMA)
- Teledesic
- telefonia celular
- transferência (*handoff*)
- travessia dupla (*double crossing*)
- triangulação
- trilateração
- Vetor de Alocação de Rede (NAV – Network Allocation Vector)

6.4.3 Resumo

As LANs sem fio foram formalizadas com o padrão IEEE 802.11, que define dois serviços: Conjunto Básico de Serviços (BSS – Basic Service Set) e Conjunto Estendido de Serviços (ESS – Extended Service Set). O método de acesso usado na subcamada MAC denominada Função de Coordenação Distribuída (DCF – Distributed Coordination Function) é o CSMA/CA. O método de acesso usado na subcamada MAC denominado Função de Coordenação Pontual (PCF – Point Coordination Function) é a técnica de varredura. O Bluetooth é uma tecnologia de LAN sem fio que conecta dispositivos (conhecidos como *gadgets*) em uma pequena área. Uma rede Bluetooth é denominada uma *piconet*. O WiMAX é uma rede de acesso sem fio, que no futuro pode substituir redes DSL e a cabo.

A telefonia celular permite a comunicação entre dois dispositivos. É possível que ambos sejam móveis ou que apenas um deles o seja. Uma área de serviço celular é dividida em células. O Sistema Avançado de Telefonia Móvel (AMPS – Advanced Mobile Phone System) é um sistema de telefonia celular da primeira geração. O AMPS Digital (D-AMPS – Digital AMPS) é um sistema de telefonia celular da segunda geração que consiste em uma versão digital do AMPS. O Sistema Global para Comunicações Móveis (GSM – Global System for Mobile Communication) é um sistema de telefonia celular da segunda geração que surgiu na Europa. O Padrão Provisório 95 (IS-95 – Interim Standard 95) é um sistema de telefonia celular da segunda geração baseado nas tecnologias CDMA e DSSS. Os sistemas de telefonia celular da terceira geração fornecem comunicação pessoal universal. A quarta geração é a nova geração da telefonia celular que está se tornando popular.

Uma rede de satélites utiliza satélites para fornecer comunicação entre quaisquer pontos da Terra. Uma órbita terrestre geoestacionária (GEO – Geostationary Earth Orbit) fica no plano equatorial e gira em fase com a rotação da Terra. O Sistema de Posicionamento Global (GPS – Global Positioning System) consiste em satélites MEO (*Medium-Earth-Orbit*, ou Órbita Terrestre Média) que fornecem informações de tempo e localização para veículos e embarcações. Os satélites Iridium são satélites LEO (*Low-Earth-Orbit*, ou Órbita Terrestre Baixa) que fornecem comunicações diretas e universais de voz e de dados para terminais portáteis. Os satélites Teledesic são satélites LEO que fornecerão acesso universal à Internet de banda larga.

O IP móvel, projetado para comunicação móvel, é uma versão melhorada do protocolo IP. Uma estação móvel tem um endereço nativo (*home address*) em sua rede nativa e um endereço de tratamento (*care-of address*) na rede externa (rede visitada). Quando a estação móvel está na rede externa, um agente nativo (*home agent*) encaminha mensagens (destinadas à estação móvel) para um agente externo (*foreign agent*). Um agente externo envia as mensagens encaminhadas para a estação móvel correspondente.

6.5 ATIVIDADES PRÁTICAS

6.5.1 Testes

Diversos testes interativos relativos a este capítulo podem ser encontrados no *site* www.mhhe.com/forouzan. Recomendamos que o estudante realize os testes para verificar o seu entendimento do material apresentado antes de continuar com as atividades práticas.

Questões

6-1 Compare o meio de transmissão de uma LAN com fio com o de uma LAN sem fio considerando o ambiente atual de comunicações.

6-2 Explique por que o protocolo MAC é mais importante em LANs sem fio do que em LANs com fio.

6-3 Explique por que há mais atenuação em uma LAN sem fio do que em uma LAN com fio, ignorando o ruído e a interferência.

6-4 Por que a relação sinal/ruído (denotada S/R ou SNR) em uma LAN sem fio é normalmente inferior àquela em uma LAN com fio?

6-5 O que é propagação por múltiplos caminhos? Qual é o seu efeito sobre redes sem fio?

6-6 Explique algumas das razões pelas quais o CSMA/CD não pode ser usado em uma LAN sem fio.

6-7 Liste algumas estratégias do CSMA/CA que são usadas para evitar colisões.

6-8 Em uma LAN sem fio, é atribuído à estação A um valor de IFS = 5 milissegundos e à estação B é atribuído um valor de IFS = 7 milissegundos. Qual estação tem uma prioridade maior? Explique.

6-9 Não há um mecanismo de confirmação de recepção de pacotes no CSMA/CD, porém precisamos de tal mecanismo no CSMA/CA. Explique a razão.

6-10 Para que serve o NAV no CSMA/CA?

6-11 Explique por que a fragmentação é recomendada em uma LAN sem fio.

6-12 Explique por que existe apenas um tipo de quadro em uma LAN com fio, porém quatro tipos de quadros em uma LAN sem fio.

6-13 Os endereços MAC usados no 802.3 (Ethernet com fio) e os endereços MAC usados no 802.11 (Ethernet sem fio) pertencem a dois espaços de endereços diferentes?

6-14 Um AP pode conectar uma rede sem fio a uma rede com fio. ● AP precisa ter dois endereços MAC, nesse caso?

6-15 Um AP em uma rede sem fio desempenha o mesmo papel que um *switch* da camada de

enlace em uma rede com fio. No entanto, um *switch* da camada de enlace não apresenta um endereço MAC, enquanto um AP normalmente precisa de um endereço MAC. Explique a razão para isso.

6-16 Qual é a razão pela qual o Bluetooth é normalmente denominado uma rede pessoal sem fio (WPAN) e não uma rede local sem fio (WLAN)?

6-17 Compare uma *piconet* e uma *scatternet* na arquitetura Bluetooth.

6-18 Uma *piconet* pode ter mais do que oito estações? Explique.

6-19 Qual é a largura de banda efetivamente utilizada para comunicação em uma rede Bluetooth?

6-20 Qual é o papel da camada de *rádio* no Bluetooth?

6-21 Preencha os espaços em branco. A largura de banda de 83,5 MHz do Bluetooth é dividida em ______ canais, cada um apresentando ______ MHz.

6-22 Qual é a técnica de espalhamento espectral utilizada pelo Bluetooth?

6-23 Qual é a técnica de modulação usada na camada de rádio do Bluetooth? Em outras palavras, como os dados digitais (*bits*) são transformados em sinais analógicos (ondas de rádio)?

6-24 Qual protocolo MAC é usado na camada de banda base do Bluetooth?

6-25 Qual é o papel da camada *L2CAP* no Bluetooth?

6-26 Em qual método de canalização as estações compartilham a largura de banda disponível em um canal no domínio do tempo?

a. FDMA **b.** TDMA **c.** CDMA

6-27 Em qual método de canalização as estações compartilham a largura de banda disponível em um canal no domínio das frequências?

a. FDMA **b.** TDMA **c.** CDMA

6-28 Em qual método de canalização as estações recebem códigos diferentes?

a. FDMA **b.** TDMA **c.** CDMA

6-29 Considere que dois usuários de telefonia celular utilizam FDMA para a comunicação. Se for alocada uma banda de frequências diferente para cada estação, como as duas estações podem se comunicar uma com a outra?

6-30 No TDMA, se for alocada uma parcela de tempo diferente para cada estação, como duas estações podem se comunicar uma com a outra?

6-31 No CDMA, se for alocado um código diferente para cada estação, como duas estações podem se comunicar uma com a outra?

6-32 Suponha que existem oito estações em um sistema usando CDMA. Qual é o valor dos seguintes produtos internos de códigos?

a. $c_1 \cdot c_1$ **b.** $c_1 \cdot c_4$ **c.** $c_4 \cdot c_1$

6-33 Qual é o número de sequências no CDMA para cada um dos seguintes sistemas com o número de estações dado?

a. 8 estações **b.** 12 estações **c.** 28 estações

6-34 Quantos códigos possíveis de tamanho 4 podem ser criados? Quantos deles são ortogonais?

6-35 A qual geração cada um dos seguintes sistemas de telefonia celular pertence?

a. AMPS **b.** D-AMPS **c.** IS-95

6-36 No IP móvel, o registro é necessário se a estação móvel atuar como um agente externo? Explique sua resposta.

6-37 Discuta como a mensagem ICMP de solicitação de roteador pode ser usada para a solicitação de agente.

6-38 Explique por que o pedido de registro e a resposta ao registro não são diretamente encapsulados em um datagrama IP. Por que é necessário usar um datagrama de usuário UDP?

Problemas

6-1 Em uma rede 802.11, escreva o valor do campo "endereço 1" em cada uma das seguintes situações (o *bit* à esquerda indica *Para o DS* e o *bit* à direita indica *Do DS*).

a. 00 **b.** 01 **c.** 10 **d.** 11

6-2 Em uma rede 802.11, escreva o valor do campo "endereço 2" em cada uma das seguintes situações (o *bit* à esquerda indica *Para o DS* e o *bit* à direita indica *Do DS*).

a. 00 **b.** 01 **c.** 10 **d.** 11

6-3 Em uma rede 802.11, escreva o valor do campo "endereço 3" em cada uma das seguintes situações (o *bit* à esquerda indica *Para o DS* e o *bit* à direita indica *Do DS*).

a. 00 **b.** 01 **c.** 10 **d.** 11

6-4 Em uma rede 802.11, escreva o valor do campo "endereço 4" em cada uma das seguintes situações (o *bit* à esquerda indica *Para o DS* e o *bit* à direita indica *Do DS*).

a. 00 **b.** 01 **c.** 10 **d.** 11

6-5 Em um BSS sem um AP (uma rede *ad hoc*), há cinco estações: *A*, *B*, *C*, *D* e *E*. A estação *A* precisa enviar uma mensagem para a estação *B*. Responda ás seguintes perguntas, considerando que a rede está usando o protocolo DCF:

a. Quais são os valores dos *bits Para o DS* e *Do DS* nos quadros trocados?

b. Qual estação envia o quadro RTS e quais são os valores dos campos de endereço nesse quadro?

c. Qual estação envia o quadro CTS e quais são os valores dos campos de endereço nesse quadro?

d. Qual estação envia o quadro de dados e quais são os valores dos campos de endereço nesse quadro?

e. Qual estação envia o quadro de ACK e quais são os valores dos campos de endereço nesse quadro?

6-6 Na Figura 6.61, duas redes sem fio, BSS1 e BSS2, estão conectadas por meio de um Sistema de Distribuição (DS – Distribution System) com fio, uma LAN Ethernet. Considere que a estação A no BSS1 precisa enviar um quadro de dados para a estação C no BSS2. Mostre o valor dos endereços nos quadros 802.11 e 802.3 para três transmissões: da estação A para o AP1, do AP1 para o AP2 e do AP2 para a estação C. Perceba que a comunicação entre o AP1 e o AP2 ocorre em um ambiente com fio.

6-7 Repita o problema anterior (Figura 6.61), mas considerando que o DS também é sem fio. O AP1 está conectado ao AP2 por meio de um canal sem fio. Mostre o valor dos endereços em todas as seções da comunicação: da estação A para o AP1, do AP1 para o AP2 e do AP2 para a estação C.

6-8 Considere que um quadro se desloca de uma rede com fio usando o protocolo 802.3 para uma rede sem fio usando o protocolo 802.11. Mostre como os valores dos campos do quadro 802.11 são preenchidos com os valores do quadro 802.3. Considere que a transformação ocorre no AP que está na fronteira entre as duas redes.

6-9 Considere que um quadro se desloca de uma rede sem fio usando o protocolo 802.11 para uma rede com fio usando o protocolo 802.3. Mostre como os valores dos campos do quadro 802.3 são preenchidos com os valores do quadro 802.11. Considere que a transformação ocorre no AP que está na fronteira entre as duas redes.

6-10 Considere que duas redes sem fio 802.11 estejam conectadas ao restante da Internet por meio de um roteador, conforme mostra a Figura 6.62. O roteador recebeu um datagrama IP cujo endereço de destino é 24.12.7.1, e precisa enviá-lo para a estação sem fio correspondente. Explique o processo e descreva como os valores dos campos *endereço 1*, *endereço 2*, *endereço 3* e *endereço 4* (ver Figura 6.62) são determinados nesse caso.

6-11 Na Figura 6.62 (problema anterior), considere que a estação cujo endereço IP é 24.12.10.3 precisa enviar um datagrama IP para a estação cujo endereço IP é 128.41.23.12, a qual

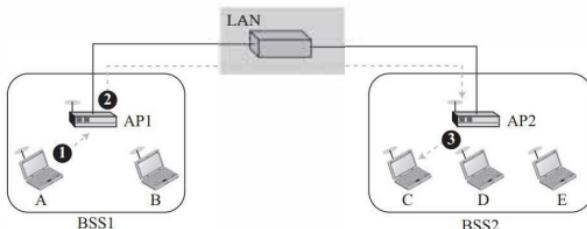



Figura 6.61 Esquema para o Problema P6-6.

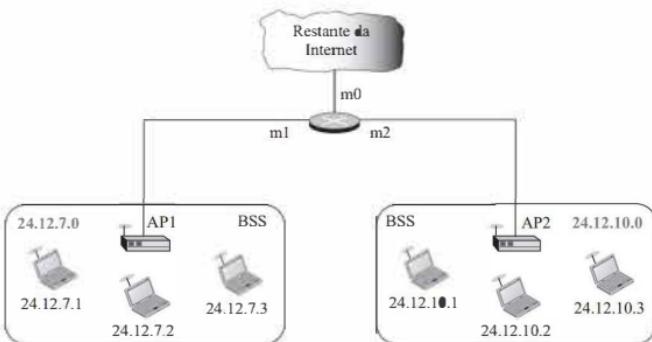



Figura 6.62 Esquema para o Problema P6-10.

se encontra em algum lugar do mundo (não mostrado na figura). Explique o processo e mostre como os valores dos campos *endereço 1*, *endereço 2*, *endereço 3* e *endereço 4* (ver Figura 6.62) são determinados nesse caso.

6-12 Um identificador de BSS (BSSID) é um endereço de 48 *bits* atribuído a um BSS em uma rede 802.11. Pesquise e descubra qual é o uso do BSSID e como os BSSIDs são atribuídos em redes *ad hoc* e infraestruturadas.

6-13 Pesquise e descubra como o controle de fluxo e de erros é feito em uma rede 802.11 usando a subcamada MAC DCF.

6-14 Em uma comunicação 802.11, o tamanho da carga útil (o corpo do quadro) é de 1.200 *bytes*. A estação decide dividir o quadro em três fragmentos, cada um contendo 400 *bytes* da carga útil. Responda às seguintes perguntas:

- **a.** Qual seria o tamanho do quadro de dados se a fragmentação não fosse feita?
- **b.** Qual é o tamanho de cada quadro após a fragmentação?
- **c.** Quantos *bytes* no total são enviados após a fragmentação (ignorando os quadros de controle extras)?
- **d.** Quantos *bytes* extras são enviados por causa da fragmentação (novamente, ignorando quadros de controle extras)?

6-15 Tanto o protocolo IP como o projeto 802.11 fragmentam seus pacotes. O IP fragmenta um datagrama na camada de rede; o 802.11 fragmenta um quadro na camada de enlace de dados. Discuta semelhanças e diferenças entre os dois esquemas de fragmentação, considerando os diferentes campos e subcampos utilizados em cada protocolo.

6-16 Em uma rede 802.11, suponha que a estação A tem quatro fragmentos para enviar à estação B. Se o número de sequência escolhido para o primeiro fragmento for 3273, quais são os valores do indicador de *mais fragmentos*, do *número do fragmento* e do *número de sequência*?

6-17 Em uma rede 802.11, a estação A envia um quadro de dados (não fragmentado) para a estação B. Qual o valor do campo *D* (em microssegundos) que precisa ser usado para definir o período de NAV em cada um dos seguintes quadros: RTS, CTS, dados e ACK? Considere que o tempo de transmissão para os quadros de RTS, CTS e ACK seja de 4 μs cada. O tempo de transmissão para o quadro de dados é de 40 μs e a duração do SIFS é ajustada em 1 μs. Ignore o tempo de propagação. Perceba que cada quadro deve empregar uma duração do NAV correspondente ao restante do tempo que o meio físico precisa ser reservado para completar a transação.

6-18 Em uma rede 802.11, a estação A envia dois fragmentos de dados para a estação B. Qual é o valor do campo *D* (em microssegundos) que precisa ser usado para definir o período de NAV em cada um dos seguintes quadros: RTS, CTS, dados e ACK? Considere que o tempo de transmissão para os quadros de RTS, CTS e ACK seja de 4 μs cada. O tempo de transmissão de cada fragmento é de 20 μs e a duração do SIFS é ajustada em 1 μs.

Ignore o tempo de propagação. Perceba que cada quadro deve empregar uma duração do NAV correspondente ao restante do tempo que o meio físico precisa ser reservado para completar a transação.

6-19 Em uma rede 802.11, três estações (*A*, *B* e *C*) estão disputando pelo acesso ao meio. A janela de contenção para cada estação tem 31 parcelas de tempo. A estação *A* aleatoriamente seleciona a primeira parcela; a estação *B* fica com a quinta parcela; a estação C fica com a vigésima primeira parcela. Mostre o procedimento que cada estação deve seguir.

6-20 Em uma rede 802.11, há três estações, *A*, *B* e *C*. A estação *C* está escondida de *A*, mas pode ser vista por *B* (encontra-se na área de alcance do seu rádio). Considere, então, que a estação *A* queira enviar dados para a estação *B*. Como *C* está escondida de *A*, o quadro RTS não consegue chegar até *C*. Explique como a estação *C* pode descobrir que o canal está bloqueado por *A* e que ela deve se abster de transmitir dados.

6-21 Uma rede 802.11 pode usar quatro Espaços entre Quadros (IFSs – Interframe Spaces) para atrasar a transmissão de um quadro em diferentes situações. Isto permite que as transmissões de baixa prioridade aguardem as transmissões de alta prioridade quando o canal ficar livre. Normalmente, quatro IFSs distintos são usados em aplicações diferentes, conforme mostra a Figura 6.63. Explique a utilidade desses IFSs (você pode precisar pesquisar essa informação na Internet).

6-22 Embora um quadro RTS defina o valor do tempo no qual o NAV ficará efetivo para o restante da sessão, por que o projeto 802.11 determina que outros quadros usados na sessão devem redefinir o valor do período restante do NAV?

6-23 A Figura 6.24 mostra o formato do quadro na camada-base do Bluetooth (802.15). Considerando esse formato, responda às seguintes perguntas:

a. Qual é o tamanho do intervalo de endereços em uma rede Bluetooth?

b. Quantas estações podem estar ativas ao mesmo tempo em uma *piconet* com base no valor obtido no item anterior?

6-24 Verifique se o seguinte conjunto de fichas pode pertencer a um sistema ortogonal.

$$[+1, +1] \quad \text{e} \quad [+1, -1]$$

6-25 Verifique se o seguinte conjunto de fichas pode pertencer a um sistema ortogonal.

$$[+1, +1, +1, +1],\ [+1, -1, -1, +1],\ [-1, +1, +1, -1],\ [+1, -1, -1, +1]$$

6-26 Alice e Bob estão realizando experimentos com o CDMA usando uma tabela de Walsh W2 (ver Figura 6.34). Alice usa o código [+1, +1] e Bob usa o código [+1, –1]. Considere que eles enviem um dígito hexadecimal um para o outro simultaneamente. Alice envia $(6)_{16}$ e Bob envia $(B)_{16}$. Mostre como eles podem detectar o dado que a outra pessoa enviou.

6-27 Desenhe um padrão de células com um fator de reuso de frequências de 5.

6-28 Determine a eficiência do protocolo AMPS em termos do número de chamadas simultâneas por mega-hertz de largura de banda. Em outras palavras, determine o número de chamadas que podem ser feitas com uma alocação de banda de 1 MHz.

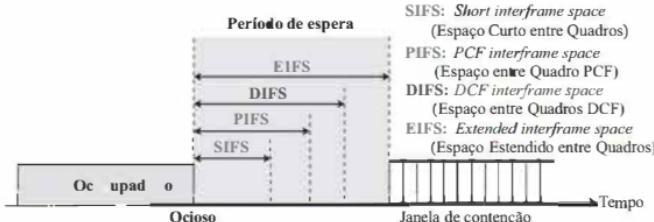



Figura 6.63 Esquema para o Problema P6-21.

6-29 Determine a eficiência do protocolo D--AMPS em termos do número de chamadas simultâneas por mega-hertz de largura de banda. Em outras palavras, determine o número de chamadas que podem ser feitas com uma alocação de banda de 1 MHz.

6-30 Determine a eficiência do protocolo GSM em termos do número de chamadas simultâneas por mega-hertz de largura de banda. Em outras palavras, determine o número de chamadas que podem ser feitas com uma alocação de banda de 1 MHz.

6-31 Determine a eficiência do protocolo IS-95 em termos do número de chamadas simultâneas por mega-hertz de largura de banda. Em outras palavras, determine o número de chamadas que podem ser feitas com uma alocação de banda de 1 MHz.

6-32 Use a lei de Kepler para verificar a exatidão de um dado valor de período e de altitude para satélites GPS.

6-33 Use a lei de Kepler para verificar a exatidão de um dado valor de período e de altitude para satélites Globalstar.

6-34 Redesenhe a Figura 6.58 para o caso em que a estação móvel atua como um agente externo.

6-35 Crie uma mensagem de anúncio de agente nativo usando 1456 como o valor do número de sequência e um tempo de vida de três horas. Use os valores que desejar para os *bits* no campo de código. Calcule e insira o valor do campo de comprimento.

6-36 Crie uma mensagem de anúncio de agente externo usando 1672 como o valor do número de sequência e um tempo de vida de três horas. Use os valores que desejar para os *bits* no campo de código. Utilize pelo menos três endereços de tratamento de sua escolha. Calcule e insira o valor do campo de comprimento.

6-37 Temos as informações mostradas a seguir. Mostre o conteúdo do cabeçalho do datagrama IP enviado pela estação remota para o agente nativo.

> Endereço nativo (*home address*) da estação móvel: 130.45.6.7/16
>
> Endereço de tratamento (*care-of address*) da estação móvel: 14.56.8.9/8
>
> Endereço da estação remota: 200.4.7.14/24
>
> Endereço do agente nativo (*home agent*): 130.45.10.20/16
>
> Endereço do agente externo (*foreign agent*): 14.67.34.6/8

6.6 EXPERIMENTOS DE SIMULAÇÃO

6.6.1 *Applets*

Criamos alguns *applets* Java para demonstrar alguns dos principais conceitos discutidos neste capítulo. Recomendamos que o estudante ative esses *applets* no *site* www.mhhe.com/forouzan e examine cuidadosamente os protocolos em ação.

6.6.2 Experimentos de laboratório

Veja o material de apoio no *site* www.grupoa.com.br e descubra como você pode usar um simulador para LANs sem fio.

Neste laboratório, capturamos e estudamos quadros sem fio que são trocados entre uma estação sem fio e o ponto de acesso. Consulte o *site* do Grupo A para uma descrição detalhada desse experimento de laboratório.

6.7 TAREFAS DE PROGRAMAÇÃO

Escreva o código-fonte, compile e teste o programa a seguir usando a linguagem de programação de sua preferência:

Prg6-1 Escreva um programa para simular o diagrama de fluxo do CSMA/CA mostrado na Figura 6.7.

CAPÍTULO

7 CAMADA FÍSICA E MEIOS DE TRANSMISSÃO

Nossa discussão sobre a pilha de protocolos TCP/IP ficaria incompleta sem uma explicação sobre a camada física e os meios de transmissão que transportam os sinais criados por ela. Entretanto, este capítulo pode ser parcial ou totalmente ignorado se o leitor tiver algum conhecimento sobre a camada física, cujo papel, é importante ressaltar, é receber os *bits* da camada de enlace de dados e convertê-los em sinais eletromagnéticos para a transmissão. As quatro primeiras seções deste capítulo discutem essa tarefa. Depois de os *bits* terem sido convertidos em sinais, estes são enviados para os meios de transmissão, tema da nossa discussão na última seção. Dividimos este capítulo em cinco seções:

- Na primeira seção, discutimos primeiramente a relação entre dados e sinais. Mostramos, então, que os dados e sinais podem ser tanto analógicos como digitais. Em seguida, abordamos os sinais analógicos e suas características, bem como os sinais digitais e suas características. Também discutimos algumas questões relacionadas com a capacidade e o desempenho dos canais físicos.
- Na segunda seção, nos concentramos na transmissão digital. Primeiramente, mostramos como converter dados digitais em sinais digitais, e depois, dados analógicos em sinais digitais.
- Na terceira seção, nos concentramos na transmissão analógica. Primeiramente, mostramos como converter dados digitais em sinais analógicos, e depois, dados analógicos em sinais analógicos.
- Na quarta seção, primeiramente mostramos como técnicas de multiplexação podem combinar diversos sinais ocupando uma reduzida largura de banda para formar um sinal de banda larga, algo que nos permite usar um mesmo meio de transmissão para transportar diversos canais. Em seguida, explicamos como as técnicas de espalhamento spectral podem proteger um sinal contra intrusos.
- Na quinta seção, vamos abaixo da camada física e discutimos os meios de transmissão usados na comunicação de dados. Apresentamos tanto os meios guiados (fios e cabos) como meios não guiados (ar).

7.1 DADOS E SINAIS

Na camada física, a comunicação é nó a nó, porém os nós trocam sinais eletromagnéticos entre eles. A Figura 7.1 usa o mesmo cenário que mostramos nos quatro capítulos anteriores, mas a comunicação agora é na camada física.

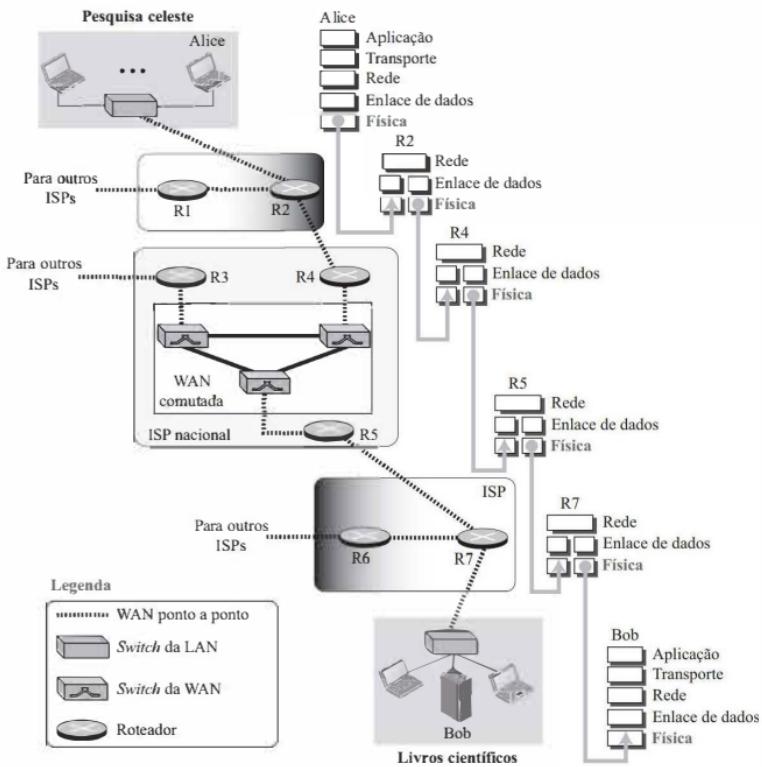



Figura 7.1 Comunicação na camada física.

Uma das principais funções da camada física é rotear *bits* entre nós. No entanto, como os *bits* representam dois valores possíveis armazenados na memória de um nó (*host*, roteador ou *switch*), eles não podem ser passados diretamente ao meio de transmissão (fios ou ar); precisam ser transformados em sinais antes da transmissão. Assim, a principal tarefa da camada física é converter de maneira eficiente esses *bits* em sinais eletromagnéticos. Primeiramente, precisamos entender a natureza dos dados, e então os tipos de sinais, para aprender a fazer essa conversão de forma eficiente.

7.1.1 Analógico e digital

Os dados podem ser analógicos ou digitais. O termo **dados analógicos** refere-se a informações contínuas. Os dados analógicos, como os sons criados por uma voz humana, assumem valores contínuos. Quando uma pessoa fala, uma onda analógica é criada no ar. Esse sinal pode ser captado por um microfone e convertido em um sinal analógico ou então amostrado e convertido em um sinal digital.

Os dados digitais assumem valores discretos. Por exemplo, os dados são armazenados na memória do computador na forma de 0s e 1s e podem ser convertidos em um sinal digital modulado ou em um sinal analógico para a transmissão através de um meio.

Assim como os dados que eles representam, os *sinais* podem ser analógicos ou digitais. Um **sinal analógico** apresenta um número infinitamente grande de níveis de intensidade em certo intervalo de tempo. À medida que a onda se move do valor A para o valor B, ela pode assumir um número infinito de valores ao longo do caminho. Um **sinal digital**, por outro lado, pode apresentar apenas um número limitado de valores bem definidos. Embora cada valor possa ser um número arbitrário, geralmente eles são simples, como 1 e 0. A forma mais simples de mostrar sinais é desenhando-os em um par de eixos perpendiculares. O eixo vertical representa o valor ou a força de um sinal. O eixo horizontal representa o tempo. A Figura 7.2 ilustra um sinal analógico e um digital.

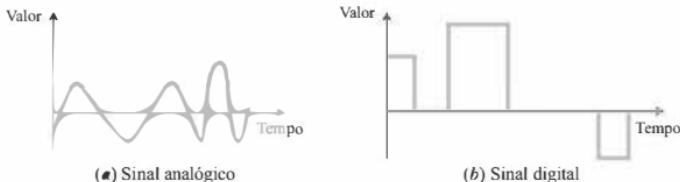



Figura 7.2 Comparação entre sinais analógicos e digitais.

Sinais analógicos

Primeiramente, explicaremos os sinais analógicos. Um sinal analógico pode assumir uma de duas formas: *periódico* ou *aperiódico* (*não periódico*). Um **sinal analógico periódico** apresenta um padrão dentro de um intervalo de tempo mensurável, denominado **período**, e repete esse padrão em intervalos de tempo idênticos posteriores. A conclusão de um padrão completo é chamada *ciclo*. Um **sinal analógico aperiódico** varia sem exibir um padrão ou ciclo que se repete ao longo do tempo. No contexto de comunicação de dados, geralmente usamos sinais analógicos periódicos.

Os sinais analógicos periódicos podem ser classificados como simples ou compostos. Um sinal analógico periódico simples consiste em uma **onda senoidal**, não podendo ser decomposto em sinais ainda mais básicos. Um sinal analógico periódico composto é uma combinação de diversas ondas senoidais. A onda senoidal é a forma mais fundamental de um sinal analógico periódico. Quando a visualizamos como uma simples curva oscilante, sua variação no decorrer de um ciclo é suave e consistente, criando um fluxo contínuo e ondulado. A Figura 7.3 mostra uma onda senoidal. Cada ciclo é constituído por um único arco acima do eixo de tempo seguido por um único arco abaixo dele.

Uma onda senoidal pode ser representada por três parâmetros: a *amplitude de pico*, a *frequência* e a *fase*. Esses três parâmetros descrevem completamente uma onda senoidal. A **amplitude de pico** de um sinal corresponde ao valor absoluto de sua intensidade mais elevada e é proporcional à energia que o sinal transporta. Para sinais elétricos, a amplitude de pico é normalmente medida em volts. A Figura 7.4 mostra dois sinais e suas amplitudes de pico. O *período* refere-se à quantidade de tempo, em segundos, que um sinal leva para completar um ciclo. A **frequência**, medida em Hertz (Hz), refere-se ao número de períodos em 1 segundo. Perceba que período e frequência referem-se a uma mesma característica definida de duas formas distintas. Período e frequência são inversos um do outro ($f = 1/T$).

O termo **fase** descreve a posição da forma de onda em relação ao instante de tempo 0. Se pensarmos na onda como algo que pode ser deslocado para trás ou para a frente ao longo do eixo do tempo, a fase define a quantidade desse deslocamento. Ela indica o estado do primeiro ciclo; é medida em graus ou radianos (360° corresponde a 2π radianos).

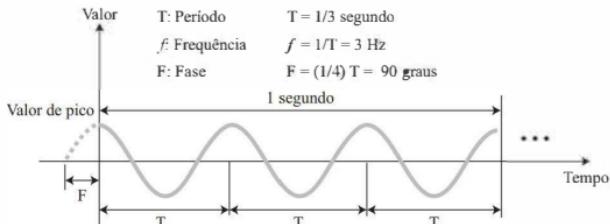



Figura 7.3 Uma onda senoidal.

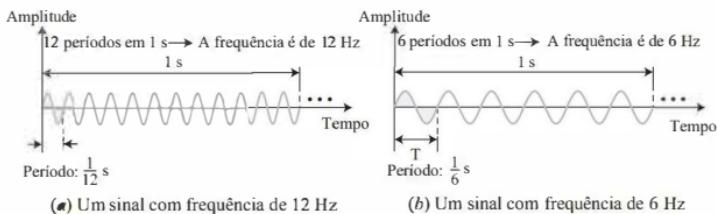



Figura 7.4 Comprimento de onda e período.

O **comprimento de onda** é outra característica de um sinal viajando por um meio de transmissão. O comprimento de onda corresponde à distância que um sinal simples pode viajar em um período. O comprimento de onda vincula o período ou a frequência de uma onda senoidal simples à velocidade de propagação do meio (ver Figura 7.4).

Embora a frequência de um sinal seja independente do meio, o comprimento de onda depende tanto da frequência como do meio. O comprimento de onda é uma propriedade de qualquer tipo de sinal. No contexto de comunicações de dados, geralmente usamos o comprimento de onda para descrever a transmissão de luz em uma fibra óptica.

O comprimento de onda pode ser calculado a partir da velocidade de propagação do meio e da frequência do sinal. Se representarmos o comprimento de onda por λ, a velocidade de propagação por c (velocidade da luz) e a frequência por f, temos

$$\lambda = c / f = c \times T$$

Domínios do tempo e da frequência

Uma onda senoidal é completamente definida por sua amplitude, frequência e fase. Até aqui temos mostrado uma onda senoidal usando a chamada **representação no domínio do tempo**, que mostra as variações na amplitude do sinal em função do tempo. Para mostrar a relação entre amplitude e frequência, podemos usar a **representação no domínio da frequência**. A Figura 7.5 mostra um sinal representado usando os domínios do tempo e da frequência.

No domínio da frequência, uma onda senoidal é representada por uma linha vertical. A posição da linha mostra a frequência; a sua altura mostra a amplitude de pico.

(*a*) Uma onda senoidal no domínio do tempo

(*b*) A mesma onda senoidal no domínio da frequência

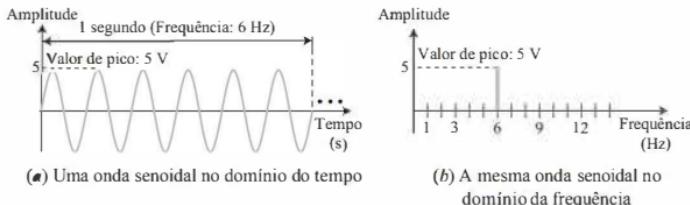



Figura 7.5 Representações no domínio do tempo e no domínio da frequência de uma onda senoidal.

Exemplo 7.1

O domínio da frequência é mais compacto e útil quando estamos lidando com mais de uma onda senoidal. Por exemplo, a Figura 7.6 mostra três ondas senoidais, cada uma com amplitude e frequência diferentes. Todas elas podem ser representadas usando três linhas verticais no domínio da frequência.

(*a*) Representação no domínio do tempo

(*b*) Representação no domínio da frequência

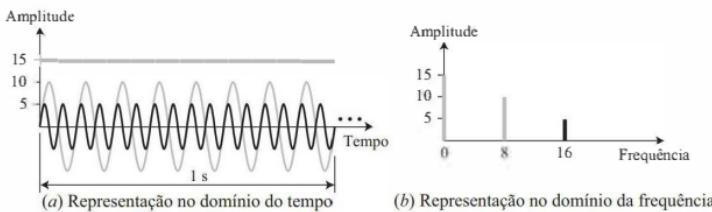



Figura 7.6 O domínio do tempo e o domínio da frequência de três ondas senoidais.

Sinais compostos

Até aqui, nos concentramos em ondas senoidais simples, que têm muitas aplicações na vida cotidiana, como para enviar energia de um lugar para outro. No entanto, se tivéssemos uma única onda senoidal para transmitir uma conversa por telefone, ela não faria sentido e não carregaria qualquer informação. Ouviríamos apenas um zumbido; precisamos enviar um sinal composto para sermos capazes de comunicar dados. Um **sinal composto** é constituído por diversas ondas senoidais simples.

No início dos anos 1900, o matemático francês Jean-Baptiste Fourier mostrou que qualquer sinal composto é, na verdade, uma combinação de ondas senoidais simples com diferentes frequências, amplitudes e fases.

Um sinal composto pode ser periódico ou aperiódico. Um sinal periódico composto pode ser decomposto em uma série de ondas senoidais simples com frequências discretas – frequências que são múltiplos inteiros da frequência fundamental (1*f*, 2*f*, 3*f* e assim por diante). Um sinal composto aperiódico pode ser decomposto em uma combinação de um número infinito de ondas senoidais simples com frequências contínuas, ou seja, frequências que assumem valores no domínio dos números reais.

Largura de banda

A faixa de frequências contidas em um sinal composto determina sua **largura de banda**, que é normalmente, uma diferença entre dois números. Por exemplo, se um sinal composto contém frequências entre 1.000 e 5.000, a sua largura de banda é 5.000 – 1.000, ou 4.000.

A largura de banda de um sinal composto é a diferença entre a maior e a menor frequências contidas naquele sinal.

A Figura 7.7 ilustra o conceito de largura de banda e mostra dois sinais compostos, um periódico e outro aperiódico. A largura de banda do sinal periódico contém diversas frequências discretas cujos valores são múltiplos inteiros da frequência fundamental. A largura de banda do sinal aperiódico apresenta a mesma faixa de valores, mas as frequências são contínuas.

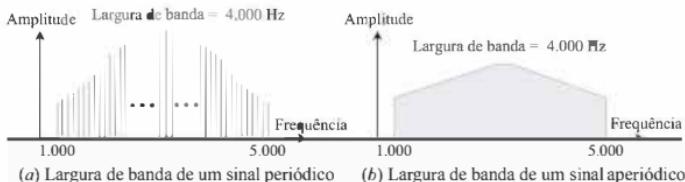



Figura 7.7 A largura de banda de sinais compostos periódicos e aperiódicos.

Sinais digitais

Além de ser representada por um sinal analógico, a informação pode também ser representada por um sinal digital. Por exemplo, um *bit* 1 pode ser codificado como uma tensão positiva e um *bit* 0 como uma tensão nula. Um sinal digital pode ter mais de dois níveis. Neste caso, podemos enviar mais de 1 *bit* com cada nível. A Figura 7.8 mostra dois sinais, um com dois níveis e outro com quatro.

Enviamos 1 *bit* por nível na parte (*a*) da figura e 2 *bits* por nível na parte (*b*) da figura. De forma geral, se um sinal tem L níveis, cada nível pode carregar $\log_2 L$ *bits*.

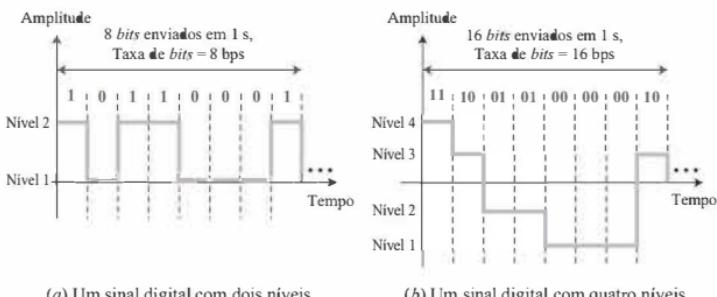



Figura 7.8 Dois sinais digitais: um com dois níveis de sinal e o outro com quatro níveis de sinal.

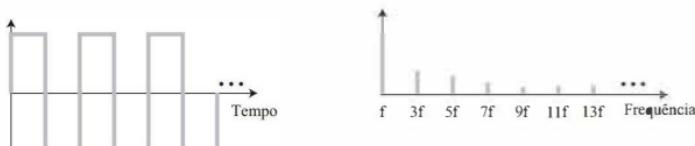
Taxa de *bits*

A maioria dos sinais digitais são aperiódicos e, portanto, o período e frequência não são características apropriadas para descrevê-los. Outro termo – *taxa de bits* (em vez de *frequência*) – é usado para descrever sinais digitais. A **taxa de *bits*** corresponde ao número de *bits* enviados em 1 segundo, expressado em *bits* por segundo (bps).

Exemplo 7.2

Considere que precisamos receber documentos de texto a uma taxa de 100 páginas por minuto. Qual é a taxa de *bits* necessária para o canal? Uma página tem em média 24 linhas com 80 caracteres em cada uma. Se considerarmos que um caractere requer 8 *bits*, a taxa de *bits* será

$$100 \times 24 \times 80 \times 8 = 1.536.000 \text{ bps} = 1.536 \text{ Mbps}.$$

Comprimento de *bit*

Discutimos o conceito de comprimento de onda para um sinal analógico: a distância que um ciclo ocupa no meio de transmissão. Podemos definir algo semelhante para um sinal digital: o comprimento de *bit*. O **comprimento de *bit*** corresponde à distância que um *bit* ocupa no meio de transmissão.

Comprimento de *bit* = velocidade de propagação × duração do *bit*

Sinal digital como um sinal analógico composto

De acordo com a análise de Fourier, um sinal digital corresponde a um sinal analógico composto. A largura de banda é infinita, como você já deve ter suposto. Podemos, intuitivamente, chegar a esse conceito quando consideramos um sinal digital, que, no domínio do tempo, corresponde a segmentos de linha verticais e horizontais conectados. Uma linha vertical no domínio do tempo significa uma frequência de valor infinito (variação súbita em um curto intervalo de tempo); uma linha horizontal no domínio do tempo significa uma frequência de valor zero (sem variação no tempo). Ir de uma frequência zero até outra infinita (e vice-versa) implica que todas as frequências intermediárias fazem parte do domínio.

A análise de Fourier pode ser usada para decompor um sinal digital. Se este for periódico, algo raro em comunicações de dados, a representação do sinal decomposto no domínio da frequência apresenta uma largura de banda infinita e frequências discretas. Se for aperiódico, o sinal decomposto ainda apresenta uma largura de banda infinita, mas as frequências são contínuas. A Figura 7.9 mostra um sinal digital periódico e um aperiódico juntamente com suas larguras de banda. Observe que ambas as larguras de banda são infinitas, mas o sinal periódico apresenta frequências discretas, enquanto no sinal aperiódico, as frequências são contínuas.

Figura 7.9 Os domínios de tempo e de frequência de sinais digitais periódicos e aperiódicos.

Transmissão de sinais digitais

Na discussão anterior, dissemos que um sinal digital, periódico ou aperiódico, é um sinal composto analógico com frequências entre zero e infinito. A partir de agora, consideraremos o caso de um sinal digital aperiódico, similar àqueles que encontramos em comunicações de dados. A questão fundamental é: como podemos enviar um sinal digital do ponto *A* ao ponto *B*? Podemos transmitir um sinal digital usando uma dentre duas abordagens distintas: transmissão em banda base ou transmissão em banda larga (utilizando modulação).

Transmissão em banda base

A transmissão em banda base corresponde ao envio de um sinal digital por um canal sem transformar o sinal digital em analógico. A Figura 7.10 mostra a **transmissão em banda base**.

Figura 7.10 Transmissão em banda base.

A transmissão em banda base requer o uso de um **canal passa-baixas**, um canal com uma banda que se inicie em zero. Esse é o caso se tivermos um meio dedicado cuja largura de banda seja usada por um único canal. Por exemplo, toda a largura de banda de um cabo que conecta dois computadores corresponde a um único canal. Como outro exemplo, podemos conectar diversos computadores a um meio, porém não permitir que mais de duas estações se comuniquem ao mesmo tempo. Temos novamente um canal passa-baixas, que pode ser usado para comunicações em banda base.

Exemplo 7.3

Um exemplo de um canal dedicado no qual a largura de banda inteira do meio é usada como um único canal é uma LAN. Quase todas as LANs com fios atuais utilizam um canal dedicado para que duas estações se comuniquem umas com as outras.

Transmissão em banda larga

A transmissão em banda larga ou modulação corresponde a transformar o sinal digital em um sinal analógico para a transmissão. A modulação nos permite usar um **canal passa-faixa** – um canal com uma banda que não se inicia em zero. Esse tipo de canal é mais facilmente encontrado que um canal passa-baixas. A Figura 7.11 mostra um canal passa-faixa.

Perceba que um canal passa-baixas pode ser considerado um canal passa-faixa cuja frequência inferior se inicia em zero. A Figura 7.12 mostra a modulação de digital. Na figura, um sinal digital é convertido em analógico composto. Usamos um sinal analógico de frequência única (denominado sinal de *portadora*); a amplitude da portadora foi alterada para se parecer com o sinal digital. O resultado, no entanto, não é um sinal de frequência única; trata-se de um sinal composto, conforme veremos mais adiante. No receptor, o sinal analógico recebido é convertido em um sinal digital e o resultado é uma réplica daquilo que foi enviado.

Figura 7.11 Largura de banda de um canal passa-faixa.

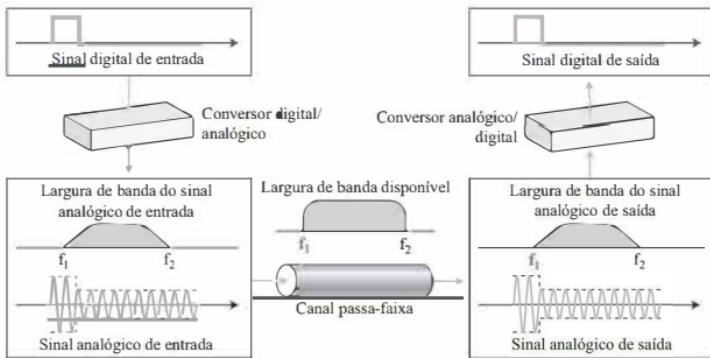



Figura 7.12 Modulação de um sinal digital para transmissão em um canal passa-faixa.

Exemplo 7.4

Um exemplo de transmissão em banda larga usando modulação é o envio de dados de um computador por meio de uma linha telefônica de assinante, a linha conectando uma residência à central telefônica. Tais linhas, instaladas há muitos anos, são projetadas para transportar voz (sinal analógico) com uma largura de banda limitada (frequências entre 0 e 4 kHz). Embora esse canal possa ser usado como um canal passa-baixas, ele é normalmente considerado um canal passa-faixa. Uma razão para isso é que a largura de banda é tão estreita (4 kHz) que, se tratarmos o canal com um passa-baixas e o usarmos para a transmissão em banda base, a taxa de *bits* máxima pode chegar a apenas 8 kbps (conforme explicado mais adiante). A solução é considerar o canal como um canal passa-faixa, converter o sinal digital proveniente do computador em um sinal analógico e enviá-lo. Podemos instalar dois conversores para transformar o sinal digital em analógico e vice-versa no lado do receptor. O conversor, nesse caso, é denominado um *modem* (*modulador/demodulador*), conforme discutido no Capítulo 5.

Exemplo 7.5

Um segundo exemplo é o telefone celular digital. Para obter uma melhor recepção, os telefones celulares digitais digitalizam a voz analógica. Embora a largura de banda alocada a uma empresa prestadora de serviços de telefonia celular digital seja muito grande, ainda assim não podemos enviar o sinal digitalizado sem conversão. A razão é que temos apenas um canal passa-faixa disponível entre o usuário que faz a chamada e aquele que a recebe. Por exemplo, se a largura de banda disponível for W e permitirmos que 1.000 pares conversem simultaneamente, o canal disponível será W/1.000, o que corresponde apenas a uma parte da largura de banda completa. Precisamos converter a voz digitalizada em um sinal analógico composto antes da transmissão.

7.1.2 Problemas na transmissão

Os sinais viajam por meios de transmissão que não são perfeitos; as imperfeições causam danos ao sinal. Isto significa que o sinal no início do meio não é igual ao do final do meio. Aquilo que é enviado não é igual àquilo que é recebido. Três causas de danos ao sinal são atenuação, distorção e ruído.

Atenuação

Atenuação significa uma perda de energia. Quando um sinal, simples ou composto, viaja por um meio, ele perde parte da sua energia para superar a resistência do meio. É por isso que um fio que transporta sinais elétricos fica morno, ou mesmo quente, depois de algum tempo. Uma parte da energia elétrica do sinal é convertida em calor. Para compensar essa perda, amplificadores são usados para aumentar o sinal. A Figura 7.13 mostra o efeito da atenuação e da amplificação.

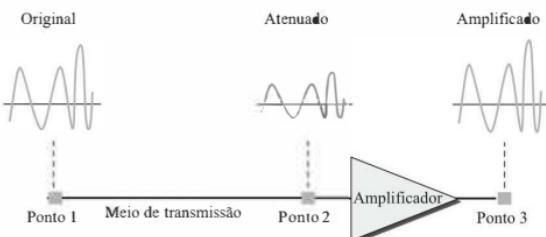



Figura 7.13 Atenuação e amplificação.

Para mostrar que um sinal perdeu ou ganhou força, os engenheiros usam o **decibel (dB)**, unidade que mede as forças relativas de dois sinais ou de um mesmo sinal em dois pontos diferentes. Perceba que o decibel é negativo se o sinal for atenuado e positivo se o sinal for amplificado.

$$\mathbf{dB = 10 \log_{10} (P_2/P_1)}$$

As variáveis P_1 e P_2 correspondem à potência de um sinal nos pontos 1 e 2, respectivamente.

Exemplo 7.6

Considere que um sinal viaja por um meio de transmissão e que a sua potência seja reduzida pela metade. Isto significa que $P_2 = 0,5\ P_1$. Nesse caso, a atenuação (perda de potência) pode ser calculada como

$$10 \log_{10} P_2/P_1 = 10 \log_{10} (0,5\ P_1)\ /P_1 = 10 \log_{10} 0,5 = 10 \times (-0,3) = -3 \text{ dB.}$$

Uma perda de 3 dB (–3 dB) é equivalente à perda de metade da potência.

Distorção

Distorção significa que a forma do sinal é alterada. A distorção pode afetar um sinal composto constituído por diferentes frequências. Cada componente do sinal apresenta sua própria velocidade de propagação em um meio e, portanto, o seu próprio atraso até chegar ao destino final. Diferenças no atraso podem criar uma diferença de fase se o valor do atraso não for exatamente igual à duração do período. Em outras palavras, os componentes do sinal que chegam no receptor apresentam fases diferentes daquelas apresentadas no remetente. A forma do sinal composto não é, portanto, a mesma. A Figura 7.14 mostra o efeito da distorção em um sinal composto.

Ruído

Ruído é uma outra fonte de problemas. Diversos tipos de ruído, como térmico, indutivo, interferências cruzadas e ruído impulsivo, podem corromper o sinal. O ruído térmico consiste no movimento

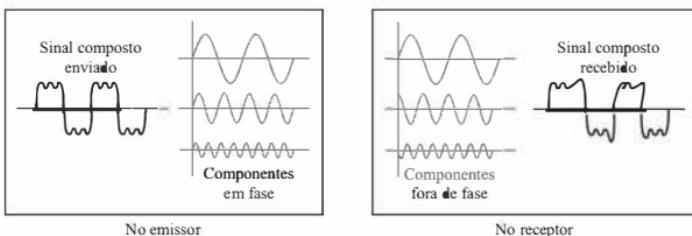



Figura 7.14 Distorção

aleatório de elétrons em um fio, o qual gera um sinal extra não enviado originalmente pelo emissor. O ruído indutivo provém de fontes como motores e eletrodomésticos. Esses dispositivos atuam como uma antena emissora, enquanto o meio de transmissão age como a antena receptora. **Interferência cruzada** (também denominada **diafonia**) corresponde ao efeito de um fio sobre o outro. Um fio atua como uma antena emissora e o outro como a antena receptora. O ruído impulsivo é um pulso (um sinal com energia elevada e duração muito curta) proveniente de linhas de transmissão de energia, raios, e assim por diante. A Figura 7.15 mostra o efeito do ruído sobre um sinal. No Capítulo 5, discutimos como detectar e controlar esses tipos de erros.

Figura 7.15 Ruído

Relação Sinal/Ruído

Conforme veremos mais adiante, para determinar o limite teórico da taxa de *bits*, precisamos saber a razão entre a potência do sinal e a potência de ruído. A **relação Sinal/Ruído (S/R)**, ou ***Signal-to-Noise Ratio*** **(SNR)**, é definida como

S/R = (potência média do sinal) / (potência média do ruído).

Precisamos considerar a potência média do sinal e a potência média do ruído porque esses valores podem mudar com o tempo. A Figura 7.16 ilustra a ideia da S/R. Uma S/R alta indica que o sinal é menos corrompido pelo ruído; uma S/R baixa indica que o sinal é mais corrompido pelo ruído. Como a S/R é a razão entre duas potências, ela é muitas vezes escrita em unidades de decibéis, S/R_{dB}, definida a seguir.

$$S/R_{dB} = 10 \log_{10} S/R$$

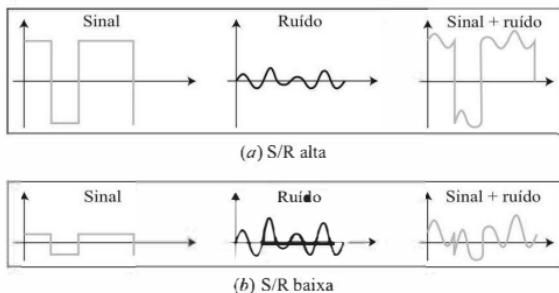



Figura 7.16 Dois casos de S/R: uma S/R alta e uma S/R baixa.

7.1.3 Limites da taxa de transferência de dados

Uma consideração muito importante na comunicação de dados é a rapidez com que podemos enviar dados, em *bits* por segundo, através de um canal. A taxa de transferência de dados depende de três fatores:

1. A largura de banda disponível
2. O número de níveis dos sinais que estamos usando
3. A qualidade do canal (o nível de ruído)

Duas fórmulas teóricas foram desenvolvidas para calcular a taxa de transferência de dados: uma por Nyquist para um canal sem ruído e outra por Shannon para um canal com ruído.

Canal sem ruído: Taxa de *bits* de Nyquist

Para um canal sem ruído, a fórmula da **taxa de *bits* de Nyquist** define o valor máximo teórico da taxa de *bits*.

$$\textbf{TaxaDeBits} = \mathbf{2 \times B \times log_2} \, L$$

Nessa fórmula, B é a largura de banda do canal, L é o número de níveis de sinal usados para representar os dados e TaxaDeBits é o número de *bits* transmitidos por segundo.

De acordo com essa fórmula, poderíamos pensar que, dada uma largura de banda específica, podemos obter qualquer taxa de *bits* que quisermos simplesmente aumentando o número de níveis de sinal. Embora a ideia esteja correta na teoria, na prática existe um limite. Quando aumentamos o número de níveis de sinal, impomos uma maior dificuldade ao receptor. Se o número de níveis de um sinal for apenas 2, o receptor pode facilmente distinguir entre um 0 e um 1. Se o número de níveis de um sinal for 64, o receptor deve ser muito mais sofisticado para distinguir entre 64 níveis diferentes. Em outras palavras, aumentar o número de níveis de um sinal reduz a confiabilidade do sistema.

Exemplo 7.7

Precisamos enviar 265 kbps usando um canal sem ruído (ideal) com uma largura de banda de 20 kHz. Quantos níveis de sinal são necessários? Podemos usar a fórmula de Nyquist, conforme mostrado a seguir:

$$265.000 = 2 \times 20.000 \times \log_2 L \quad \rightarrow \quad \log_2 L = 6,625 \quad L = 2^{6,625} = 98,7 \text{ níveis.}$$

Como esse resultado não é uma potência de 2, precisamos aumentar o número de níveis ou reduzir a taxa de *bits*. Se tivermos 128 níveis, a taxa de *bits* será de 280 kbps. Se tivermos 64 níveis, a taxa de *bits* será de 240 kbps.

Canal com ruído: Capacidade de Shannon

Na realidade, não é possível ter um canal sem ruído, pois o ruído sempre ocorre. Em 1944, Claude Shannon apresentou uma fórmula, denominada **capacidade de Shannon**, para determinar o valor teórico máximo da taxa de transferência de dados no caso de um canal ruidoso. Essa fórmula é mostrada a seguir.

$$C = B \times \log_2 (1 + S/R)$$

Nela, B é a largura de banda do canal, S/R é a relação sinal/ruído e C é a capacidade do canal em *bits* por segundo. Perceba que na fórmula de Shannon não há qualquer consideração sobre o número de níveis do sinal, o que significa que, independentemente de quantos níveis tivermos, não podemos alcançar uma taxa de transferência de dados mais alta que a capacidade do canal. Em outras palavras, a fórmula define uma característica do canal, não do método de transmissão. A capacidade define o limite superior da taxa de *bits* do canal.

Exemplo 7.8

Considere um canal extremamente ruidoso no qual o valor da relação sinal/ruído seja quase zero. Em outras palavras, o ruído é tão forte que em comparação o sinal é fraco. Para esse canal, a capacidade C é calculada conforme mostrado a seguir.

$$C = B \log_2 (1 + S/R) = B \log_2 (1 + 0) = B \log_2 1 = B \times 0 = 0$$

Isto significa que a capacidade desse canal é zero, independentemente da largura de banda. Em outras palavras, os dados são tão corrompidos que acabam sendo inúteis quando recebidos.

Exemplo 7.9

Podemos calcular o valor teórico máximo da taxa de *bits* para uma linha telefônica comum. Uma linha telefônica tem normalmente uma largura de banda de 3.000 Hz (300 a 3.300 Hz) alocada para a comunicação de dados. A relação sinal/ruído é geralmente 3.162. Para esse canal, a capacidade é calculada conforme mostrado a seguir.

$$C = B \log_2 (1 + S/R) = 3.000 \log_2 (1 + 3.162) = 34.881 \text{ bps}$$

Isto significa que a taxa de *bits* máxima para uma linha telefônica é 34.881 kbps. Se quisermos enviar dados mais rapidamente, podemos aumentar a largura de banda da linha ou melhorar a relação sinal/ruído.

Usando ambos os limites

Na prática, precisamos usar os dois métodos para determinar os limites e os níveis de sinal. Mostraremos com um exemplo.

Exemplo 7.10

Temos um canal com uma largura de banda de 1 MHz. A S/R para esse canal é 63. Quais são os valores apropriados da taxa de *bits* e de níveis de sinal?

Solução

Primeiro, usamos a fórmula de Shannon para determinar o limite superior.

$$C = B \log_2 (1 + S/R) = 10^6 \log_2 (1 + 63) = 10^6 \log_2 64 = 6 \text{ Mbps}$$

A fórmula de Shannon nos fornece o valor do limite superior, que é 6 Mbps. Para obter um melhor desempenho, escolhemos um valor mais baixo, 4 Mbps, por exemplo. Usamos, então, a fórmula de Nyquist para determinar o número de níveis de sinal.

$$4 \text{ Mbps} = 2 \times 1 \text{ MHz} \times \log_2 L \quad \rightarrow \quad \log_2 L = 2 \quad \rightarrow \quad L = 4$$

A capacidade de Shannon nos fornece o limite superior;
a fórmula de Nyquist nos diz quantos níveis de sinal são necessários.

7.1.4 Desempenho

Até aqui, discutimos as ferramentas de transmissão de dados (sinais) em uma rede e como os dados se comportam. Uma questão importante no contexto de redes refere-se à qualidade do desempenho da rede. Discutimos em detalhes qualidade do serviço, uma medida global do desempenho da rede, no Capitulo 8.

Largura de banda

Uma característica que mede o desempenho da rede é a largura de banda. No entanto, o termo pode ser usado em dois contextos diferentes com dois valores de medição distintos: largura de banda em hertz e largura de banda em *bits* por segundo.

Largura de banda em hertz

Já discutimos esse conceito. A largura de banda em hertz consiste na faixa de frequências contidas em um sinal composto ou na faixa de frequências que um canal pode deixar passar. Por exemplo, podemos dizer que a largura de banda de uma linha telefônica comum é de 4 kHz.

Largura de banda em *bits* por segundo

O termo *largura de banda* também pode se referir ao número de *bits* por segundo que pode ser transmitido por um canal, por um enlace, ou mesmo por uma rede. Por exemplo, podemos dizer que a largura de banda de uma rede *Fast Ethernet* (ou dos enlaces nessa rede) é de 100 Mbps. Isto significa que essa rede pode enviar dados a uma velocidade de 100 Mbps.

Relação

Existe uma relação explícita entre a largura de banda em hertz e a largura de banda em *bits* por segundo. Basicamente, um aumento na largura de banda em hertz leva a um aumento na largura de banda em *bits* por segundo. A relação exata depende se estamos usamos transmissão em banda base ou transmissão em banda larga.

Exemplo 7.11

A largura de banda de uma linha telefônica comum é de 4 kHz para voz ou dados. A largura de banda dessa linha para a transmissão de dados pode chegar a 56 kbps usando um modem sofisticado para transformar o

sinal digital em analógico. Se a companhia telefônica melhorar a qualidade da linha e aumentar a largura de banda para 8 kHz, podemos enviar dados a 112 kbps.

Vazão

A **vazão** é uma medida da velocidade com que podemos de fato enviar dados por uma rede. Embora, à primeira vista, a largura de banda em *bits* por segundo e a vazão pareçam grandezas equivalentes, elas são diferentes. Um enlace pode ter uma largura de banda de *B* bps, porém podemos enviar apenas *T* bps por ele, onde *T* é sempre menor que *B*. Em outras palavras, a largura de banda é uma medida do potencial de um enlace; a vazão é uma medida da rapidez com que podemos efetivamente enviar dados. Por exemplo, podemos ter um enlace com uma largura de banda de 1 Mbps, porém os dispositivos conectados nas extremidades do enlace podem ser capazes de lidar apenas com 200 kbps. Isto significa que não podemos enviar dados a uma taxa maior que 200 kbps nesse enlace.

Considere uma rodovia projetada para permitir a passagem de 1.000 carros por minuto de um ponto a outro. Se, entretanto, houver um congestionamento na estrada, esse valor pode ser reduzido para 100 carros por minuto. A largura de banda é de 1.000 carros por minuto; a vazão é de 100 carros por minuto.

Latência (atraso)

A latência, ou atraso, define quanto tempo leva para uma mensagem inteira chegar completamente ao destino, a contar do momento em que o primeiro *bit* foi enviado pelo emissor. Já discutimos o atraso de um pacote no Capítulo 4. Nele, dissemos que normalmente temos quatro tipos de atraso: de propagação, de transmissão, de fila e de processamento. O atraso ou latência total é calculado como

Latência = atraso de propagação = atraso de transmissão = atraso de fila = atraso de processamento.

Produto largura de banda-atraso

Largura de banda e atraso são duas métricas de desempenho de um enlace. Entretanto, uma medida muito importante no contexto de comunicações de dados é o produto dessas duas grandezas, conhecido como produto largura de banda-atraso. Discutiremos essa questão em mais detalhes usando dois casos hipotéticos como exemplo (Figura 7.17).

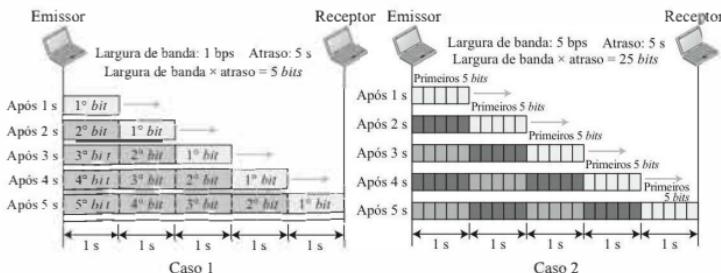



Figura 7.17 Preenchendo o enlace com *bits* nos casos 1 e 2.

- **Caso 1.** Considere que temos um enlace com uma largura de banda de 1 bps (algo pouco realista, mas bom para fins de demonstração). Suponha também que o atraso do enlace seja de 5 segundos (também pouco realista). Queremos entender o que o produto largura de banda-atraso significa nesse caso. Observando a figura, podemos ver que o produto da largura de banda pelo atraso (1 × 5) corresponde ao número máximo de *bits* que o enlace pode comportar. Em qualquer momento, não pode haver mais do que 5 *bits* no enlace.
- **Caso 2.** Agora, consideremos que temos uma largura de banda de 5 bps com um atraso de 5 segundos. A Figura 7.17 mostra que pode haver um máximo de 5 × 5 = 25 *bits* no enlace. O motivo é que, em cada segundo, existem 5 *bits* no enlace; a duração de cada *bit* é de 0,20 s.

Os dois casos anteriores mostram que o produto da largura de banda pelo atraso corresponde ao número de *bits* que pode ser comportado pelo enlace. Essa medida é importante se precisarmos enviar dados em rajadas e aguardar a confirmação de cada rajada antes de enviar a próxima. Para usar a capacidade máxima do enlace, precisamos fazer com que o tamanho de nossa rajada seja duas vezes o valor do produto largura de banda-atraso; precisamos ocupar todo o canal *full-duplex* (nos dois sentidos). O emissor deve enviar uma rajada de dados de (2 × largura de banda × atraso) *bits*. Em seguida, o emissor aguarda a confirmação da recepção de parte da rajada antes de enviar outra rajada. O valor 2 × largura de banda × atraso corresponde ao número de *bits* que podem estar em trânsito a qualquer momento.

O produto largura de banda-atraso define o número de *bits* que o enlace é capaz de comportar.

Exemplo 7.12

Podemos enxergar o enlace entre dois pontos como um tubo. A seção transversal do tubo representa a largura de banda, enquanto o comprimento do tubo representa o atraso. Podemos dizer que o volume do tubo corresponde ao produto largura de banda-atraso, conforme mostrado na Figura 7.18.

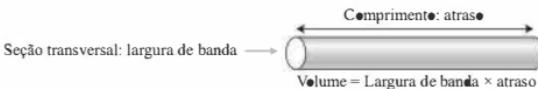



Figura 7.18 Conceito de produto largura de banda-atraso.

Jitter

Outra questão de desempenho que está relacionada ao atraso é o ***jitter***, ou **variação de atraso**. Basicamente, podemos dizer que o *jitter* é um problema se pacotes de dados diferentes sofrerem atrasos diferentes e a aplicação que usa os dados no lado do receptor for sensível a questões de tempo (dados de áudio e vídeo, por exemplo). Se o atraso para o primeiro pacote for de 20 ms, para o segundo for de 45 ms e para o terceiro for de 40 ms, então a aplicação de tempo real que usa os pacotes sofrerá com o *jitter*. Discutimos o *jitter* com mais detalhes no Capítulo 8.

7.2 TRANSMISSÃO DIGITAL

Uma rede de computadores é projetada para enviar informações de um ponto a outro. Essas informações precisam ser convertidas em um sinal digital ou analógico para a transmissão. Nesta seção,

discutimos a primeira alternativa, a conversão para sinais digitais; na próxima seção, discutimos a segunda, a conversão para sinais analógicos.

Se os dados forem digitais, precisamos usar técnicas de **conversão digital-digital**, métodos que convertem os dados digitais em sinais digitais. Se os dados forem analógicos, é preciso utilizar técnicas de **conversão analógico-digital**, métodos que transformam um sinal analógico em digital.

7.2.1 Conversão digital-digital

Já discutimos dados e sinais. Dissemos que os dados podem ser digitais ou analógicos. Também dissemos que os sinais que representam dados podem ser digitais ou analógicos. Nesta seção, vemos como podemos representar os dados digitais usando sinais digitais. A conversão envolve três técnicas: codificação de linha, codificação de bloco e embaralhamento (*scrambling*). A codificação de linha é sempre necessária; já a codificação bloco e o embaralhamento podem ou não ser necessários.

Codificação de linha

Codificação de linha consiste no processo de converter dados digitais em sinais digitais. Consideramos que os dados, na forma de texto, números, imagens gráficas, áudio ou vídeo, são armazenados na memória do computador como sequências de *bits*. A codificação de linha converte uma sequência de *bits* em um sinal digital. No emissor, os dados digitais são codificados em um sinal digital; no receptor, os dados digitais são recuperados por meio da decodificação do sinal digital. A Figura 7.19 ilustra o processo.

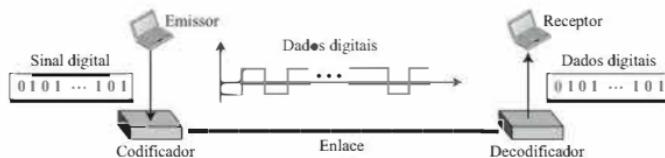



Figura 7.19 Codificação e decodificação de linha.

Podemos dividir os esquemas de codificação de linha comuns em três categorias, polares, bipolares e multiníveis, embora a literatura apresente mais categorias.

Antes de discutir cada categoria, definiremos alguns termos usados no contexto de codificação de linha. Usamos N para indicar a taxa de *bits* (número de *bits* enviados por segundo). Usamos r para indicar o número de *bits* transportados por cada alteração no nível do sinal. Também definimos S como a taxa de *bauds*, o número de alterações no nível do sinal por segundo. Normalmente, estamos interessados no valor de S_{med}, o número médio de alterações de sinal, que pode ser calculado conforme mostrado a seguir.

$$S_{med} = c \times N \times (1/r)$$

Na fórmula anterior, c é um fator relacionado com a forma como a média é calculada. Se o nível do sinal varia quando o valor do *bit* varia (de 0 para 1 ou de 1 para 0), o valor de c é 0 quando os dados são compostos apenas por 1s ou por 0s. Ele normalmente vale 1/2 na maioria dos casos.

Esquemas polares

Nos esquemas polares, o nível de tensão oscila entre um valor positivo e outro negativo, embora possa permanecer no nível zero entre esses dois valores. A Figura 7.20 mostra diversos esquemas pertencentes a essa categoria, além da largura de banda (potência do sinal *versus* a frequência), porém a frequência é normalizada (f/N).

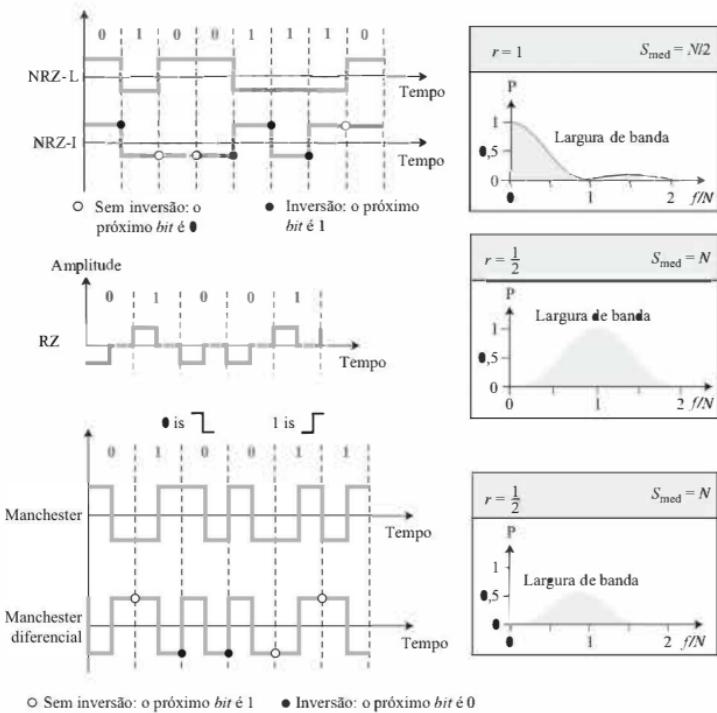



Figura 7.20 Esquemas polares.

Em esquemas **Não Retorno ao Zero** (NRZ – Non-Return-to-Zero), usamos dois níveis de amplitude para a tensão, cujo valor não permanece em zero entre eles. Existem duas versões do NRZ polar: NRZ-L e NRZ-I. No NRZ-L (NRZ-*Level*, ou NRZ-Nível), o nível da tensão define o valor do *bit*. No NRZ-I (NRZ-Inversão), a alteração e a ausência de alteração no nível da tensão determinam o valor do *bit*. Se não houver qualquer alteração, o *bit* vale 0; se houver uma alteração, o *bit* vale 1. Se houver uma longa sequência de 0s ou de 1s, a potência média do sinal fica enviesada (distorcida). No entanto, esse viés é duas vezes pior no NRZ-L. O receptor pode ter dificuldades em determinar o valor do *bit*. No NRZ-I, esse problema ocorre somente para uma longa sequência de 0s.

O principal problema com a codificação NRZ ocorre quando os relógios do emissor e do receptor não estão sincronizados. O receptor não sabe quando um *bit* terminou e o próximo *bit* está começando. Uma solução é usar um esquema **Retorno ao Zero** (RZ – Return-to-Zero), que utiliza três valores: positivo, negativo e zero. Na codificação RZ, o sinal não se altera entre *bits*, mas durante o *bit*. A principal desvantagem da codificação RZ é que ela requer duas alterações de sinal para codificar um *bit* e, portanto, ocupa uma maior largura de banda.

A ideia da codificação RZ (transições no meio do *bit*) e a ideia da codificação NRZ-L são combinadas no esquema **Manchester**. Na codificação Manchester, a duração do *bit* é dividida em duas metades. A tensão permanece em um nível na primeira metade e vai para o outro nível na

segunda metade. A transição no meio do *bit* permite a sincronização dos relógios. O **Manchester Diferencial**, por outro lado, combina as ideias do RZ e do NRZ-I. Existe sempre uma transição no meio do *bit*, mas seu valor determinado no início. Se o *bit* seguinte for 0, há uma transição; se for 1, não há transição.

Esquemas bipolares

Na codificação **bipolar** (também conhecida como *binária multinível*) existem três níveis de tensão: positivo, negativo e zero. O nível de tensão de um elemento de dados fica em zero, enquanto o nível de tensão para outros elementos alterna-se entre positivo e negativo. A Figura 7.21 mostra duas variações da codificação bipolar: AMI e pseudoternária. Um esquema comum de codificação bipolar é denominado **Inversão Alternada de Marcas** (AMI – Alternate Mark Inversion). No termo *Inversão Alternada de Marcas*, a palavra *marca* tem origem na telegrafia e significa 1. Então, AMI significa inversão alternada de 1s. Uma tensão neutra de valor zero representa o 0 binário. 1s binários são representados por tensões positivas e negativas alternadas. Em uma variação da codificação AMI, denominada **pseudoternária**, o *bit* 1 é codificado como uma tensão zero e o *bit* 0 é codificado como tensões positivas e negativas alternadas.

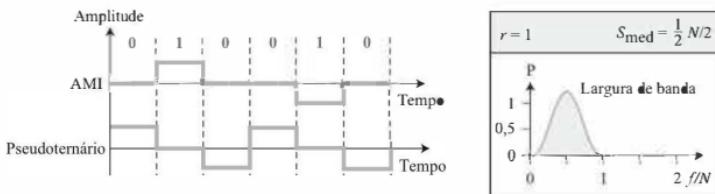



Figura 7.21 Esquemas bipolares: codificação AMI e pseudoternária.

Esquemas multinível

O desejo de aumentar a velocidade de transmissão de dados ou diminuir a largura de banda necessária resultou na criação de muitos esquemas. O objetivo é aumentar o número de *bits* por *baud* por meio da codificação de uma sequência de *m* elementos de dados em uma sequência de *n* elementos de sinal. Temos apenas dois tipos de elementos de dados (0s e 1s), o que significa que um grupo de *m* elementos de dados pode produzir uma combinação de 2^m sequências de dados. Os projetistas de códigos classificaram esse tipo de codificação como *mBnL*, onde *m* é o comprimento da sequência binária, *B* significa dados binários, *n* é o comprimento da sequência de elementos de sinal e *L* é o número de níveis usados pelo sinal. Uma letra é frequentemente usada no lugar de *L*. Por exemplo, $L = 2$ torna-se *B* (binário), $L = 3$ torna-se *T* (ternário) e $L = 4$ torna-se *Q* (quaternário). Perceba que as duas primeiras letras definem a sequência de dados, enquanto as duas últimas definem a sequência de elementos de sinal. Mostramos apenas dois desses esquemas na Figura 7.22, mas existem vários deles.

O esquema **2-Binário, 1-Quaternário** (2B1Q) utiliza sequências de dados de tamanho 2, codificando tais sequências de 2 *bits* como um elemento de sinal pertencente a um sinal de quatro níveis. Perceba que, no 2B1Q, a codificação é +1, +3, –1 e –3 para as sequências 00, 01, 10 e 11. No entanto, se o nível anterior for negativo, o valor do sinal é invertido (–1, –3, +1 e +3). Esse é o caso para a sequência 01 na figura, que foi codificada como +3 em vez de –3 devido ao nível do sinal anterior. Um esquema muito interessante é o **8-Binário, 6-Ternário** (8B6T). Esse código é utilizado em cabos 100BASE-4T, conforme vimos no Capítulo 5. A ideia é codificar uma sequência de 8 *bits* como

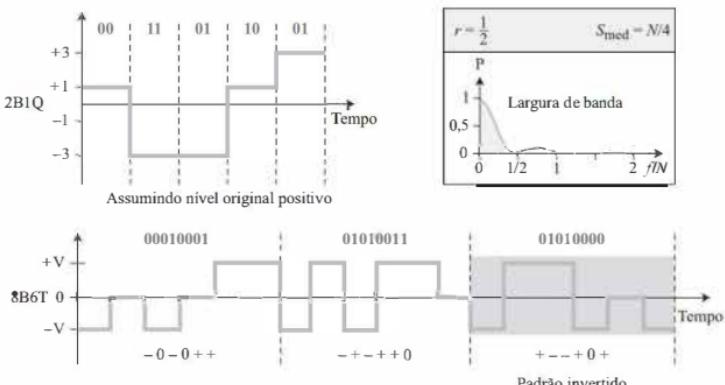



Figura 7.22 Multinível: 2B1Q e 8B6T.

uma sequência de 6 elementos de sinal, considerando um sinal de três níveis (ternário). Nesse tipo de esquema, podemos ter $2^8 = 256$ sequências diferentes de dados e $3^6 = 729$ sequências diferentes de sinais. Existem 729 – 256 = 473 elementos de sinal redundantes que fornecem sincronização e detecção de erros. Perceba que a terceira sequência é invertida para evitar um componente CC, ou seja, um componente cuja tensão média seja diferente de zero.

Codificação de bloco

Precisamos de redundância para garantir a sincronização e fornecer algum tipo de detecção de erros inerente ao código. A codificação de bloco pode prover essa redundância e um melhor desempenho do que a codificação de linha. Em geral, esquemas de **codificação de bloco** transformam um bloco de *m bits* em um bloco de *n bits*, onde *n* é maior do que *m*. A codificação de bloco é denominada uma técnica de codificação *mB/nB*. A barra no nome do esquema de codificação de bloco (por exemplo, 4B/5B) diferencia o esquema de um esquema de codificação multinível (por exemplo, 8B6T), cujo nome é escrito sem uma barra. A codificação de bloco normalmente envolve três etapas: divisão, substituição e combinação. Na etapa de divisão, uma sequência de *bits* é dividida em grupos de *m bits*. Por exemplo, na codificação 4B/5B, a sequência de *bits* original é dividida em grupos de 4 *bits*. O coração da codificação de bloco é a etapa de substituição. Nesse passo, substituímos um grupo de *m bits* por um grupo de *n bits*. Por exemplo, na codificação 4B/5B, substituímos um código de 4 *bits* por um grupo de 5 *bits*. Finalmente, os grupos de *n bits* são combinados para formar um fluxo de *bits*. O novo fluxo tem um número maior de *bits* do que o original. A Figura 7.23 ilustra o procedimento.

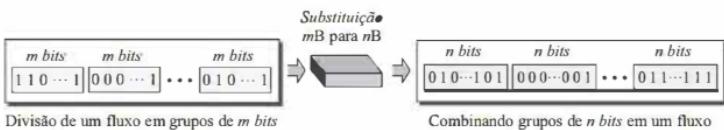



Figura 7.23 Conceito de codificação de bloco.

Codificação 4B/5B

O esquema de codificação **4-Binário/5-Binário** (4B/5B) foi projetado para ser usado em conjunto com esquemas de codificação de linha, como o NRZ-I, no qual uma sequência longa de 0s pode fazer com que o relógio do receptor perca sincronismo. Uma solução é modificar o fluxo de *bits* antes da codificação com o NRZ-I, de modo que ele não apresente uma longa sequência de 0s. O esquema 4B/5B permite que isso seja feito. O fluxo não apresenta mais do que três 0s consecutivos após a codificação de bloco. No receptor, o sinal digital codificado usando NRZ-I primeiramente é decodificado em um fluxo de *bits* e, em seguida, decodificado para remover as redundâncias. A Figura 7.24 ilustra essa ideia.

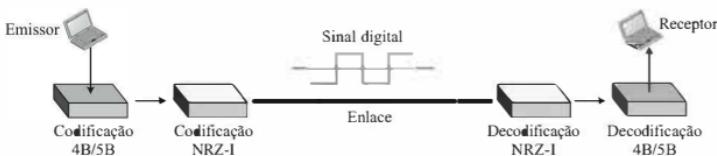



Figura 7.24 Usando codificação de bloco 4B/5B em conjunto com o esquema de codificação de linha NRZ-I.

No 4B/5B, a saída de 5 *bits* que substitui a entrada de 4 *bits* não apresenta mais do que um *bit* zero inicial (*bit* à esquerda), nem mais do que dois zeros finais (*bits* à direita). Então, quando grupos distintos são combinados para criar uma nova sequência, nunca há mais do que três 0s consecutivos.

Codificação 8B/10B

A codificação **8-Binário/10-Binário** (8B/10B) é semelhante à codificação 4B/5B exceto pelo fato de que um grupo de 8 *bits* de dados é agora substituído por um código de 10 *bits*. Esse esquema fornece maior capacidade de detecção de erros do que o 4B/5B. A codificação de bloco 8B/10B é, na realidade, uma combinação das codificações 5B/6B e 3B/4B, conforme mostra a Figura 7.25.

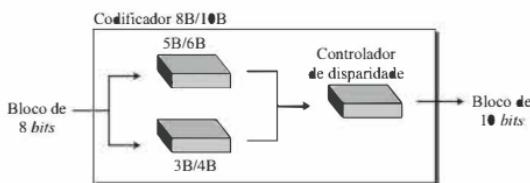



Figura 7.25 Codificação de bloco 8B/10B.

Os cinco *bits* mais significativos de um bloco de 10 *bits* passam pelo codificador 5B/6B; os três *bits* menos significativos passam por um codificador 3B/4B. A separação é feita para simplificar a tabela de mapeamento. Para prevenir o aparecimento de um longo intervalo de 0s ou 1s consecutivos, o código usa um controlador de disparidade que monitora um número excessivo de 0s em relação ao número de 1s (ou de 1s em relação a 0s). Se os *bits* do bloco sendo processado criarem uma disparidade que contribui com a disparidade anterior (em qualquer direção), cada *bit* no código é complementado (um 0 é transformado em um 1 e um 1 é transformado em um 0). A codificação

apresenta $2^{10} - 2^8 = 768$ grupos redundantes que podem ser utilizados para verificação de disparidade e detecção de erros. Em geral, a técnica é superior ao 4B/5B devido à melhor capacidade embutida de verificação de erros e à melhor sincronização.

Embaralhamento

Esquemas bifásicos adequados para enlaces dedicados entre estações em uma LAN não são adequados para comunicações de longa distância porque eles exigem uma ampla largura de banda. A combinação da codificação de bloco com a codificação de linha NRZ também não é adequada para a codificação de longa distância devido ao componente CC (um sinal com uma frequência igual a 0). A codificação bipolar AMI, por outro lado, tem uma largura de banda estreita e não cria um componente CC. Entretanto, uma longa sequência de 0s dificulta a sincronização. Se pudermos encontrar uma maneira de evitá-la no fluxo original, então podemos usar o esquema bipolar AMI para longas distâncias. Estamos em busca de uma técnica que não aumente o número de *bits* e que forneça sincronização. Em outras palavras, estamos buscando uma solução que substitui longos pulsos de nível zero por uma combinação de outros níveis, de modo a fornecer sincronização. Uma solução é a técnica conhecida como ***scrambling***, ou **embaralhamento**. Modificamos parte das regras do esquema AMI para incluir embaralhamento, conforme mostra a Figura 7.26. Perceba que o embaralhamento, ao contrário da codificação de blocos, é executado ao mesmo tempo em que a codificação. O sistema precisa inserir os pulsos necessários com base nas regras de embaralhamento definidas. Duas técnicas comuns de embaralhamento são o B8ZS e o HDB3.

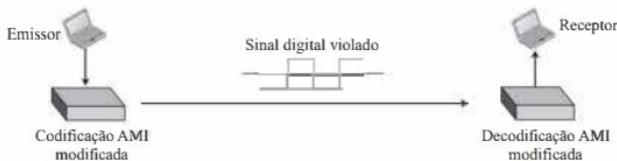



Figura 7.26 AMI usado com embaralhamento.

Codificação B8ZS

A codificação **Bipolar com Substituição de 8 Zeros** (B8ZS – Bipolar with 8-Zero Substitution) é geralmente utilizada na América do Norte. Nessa técnica, oito valores consecutivos de tensão no nível zero são substituídos pela sequência **000VB0VB**. O V nessa sequência denota *violação*, uma tensão diferente de zero que quebra uma regra da codificação AMI (uso de polaridade inversa à anteriormente usada). O B na sequência denota *bipolar*, o que significa uma tensão diferente de zero de acordo com a regra da codificação AMI. Dois casos são possíveis, conforme mostra a Figura 7.27.

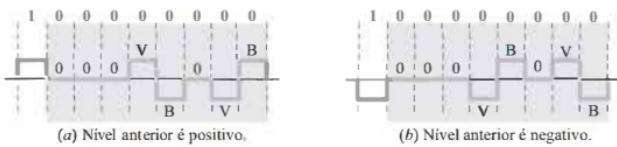



Figura 7.27 Dois casos da técnica de embaralhamento B8ZS.

Perceba que o embaralhamento, nesse caso, não altera a taxa de *bits*. Além disso, a técnica equilibra os níveis de tensão positivos e negativos (dois positivos e dois negativos), o que significa que o equilíbrio CC é mantido. Perceba também que a substituição pode alterar a polaridade de um *bit* 1 porque, depois da substituição, a codificação AMI precisa seguir suas regras.

Vale a pena mencionar mais um ponto. A letra V (violação) ou B (bipolar) aqui é relativa. O V significa a mesma polaridade que a polaridade do pulso anterior diferente de zero; B significa a polaridade oposta à polaridade do pulso anterior diferente de zero.

Codificação HDB3

O esquema **Bipolar de Alta Densidade de 3 Zeros** (HDB3 – High-Density Bipolar 3-Zero) é geralmente usado fora da América do Norte. Nessa técnica, mais conservadora que o esquema B8ZS, quatro tensões consecutivas no nível zero são substituídas por uma sequência **000V** ou **B00V**. A razão para haver duas substituições diferentes é manter o número par de pulsos diferentes de zero após cada substituição. As duas regras podem ser descritas como se segue. Se o número de pulsos diferentes de zero após a última substituição for ímpar, a sequência de substituição será **000V**, fazendo com que o número total de pulsos diferentes de zero seja par. Se o número de pulsos diferentes de zero após a última substituição for par, a sequência de substituição será **B00V**, o que também faz com que o número total de pulsos diferentes de zero seja par. A Figura 7.28 mostra um exemplo.

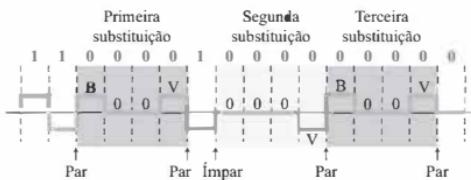



Figura 7.28 Diferentes situações na técnica de embaralhamento HDB3.

Há vários pontos que precisamos mencionar aqui. Em primeiro lugar, antes da primeira substituição, o número de pulsos diferentes de zero é par, então a primeira substituição é feita usando B00V. Feito isso, a polaridade do *bit* 1 é alterada porque o esquema AMI, após cada substituição, deve seguir suas próprias regras. Depois desse *bit*, precisamos de outra substituição, feita usando 000V porque tivemos apenas um pulso diferente de zero (ímpar) após a substituição passada. A terceira substituição é feita usando B00V porque não houve pulsos diferentes de zero após a segunda substituição (par).

7.2.2 Conversão analógico-digital

As técnicas descritas na Seção 7.2.1 convertem dados digitais em sinais digitais. Algumas vezes, porém, temos um sinal analógico, como um sinal criado por um microfone ou por uma câmera. A tendência atualmente é transformar um sinal analógico em dados digitais porque o sinal digital é menos suscetível a ruído. Nesta seção descrevemos duas técnicas, denominadas PCM e modulação delta. Depois que os dados digitais tenham sido criados (processo de digitalização), podemos usar uma das técnicas descritas na seção anterior para converter os dados digitais em um sinal digital.

Modulação por Código de Pulsos

A técnica mais comum para transformar um sinal analógico em dados digitais (processo de **digitalização**) é a denominada **Modulação por Código de Pulsos** (PCM – Pulse Code Modulation). Um codificador PCM executa três processos, conforme mostra a Figura 7.29.

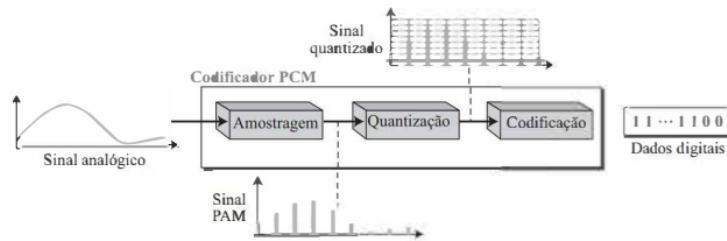



Figura 7.29 Componentes de um codificador PCM.

Os três processos são os seguintes:

1. O sinal analógico é amostrado.
2. O sinal amostrado é quantizado.
3. Os valores quantizados são codificados como sequências de *bits*.

Amostragem

O primeiro passo na PCM consiste na **amostragem**. O sinal analógico é amostrado a cada T_s segundos, onde T_s é o intervalo ou período de amostragem. O inverso do intervalo de amostragem é denominado **taxa de amostragem ou frequência de amostragem** e é denotado por f_s, onde $f_s = 1/T_s$. Existem três métodos de amostragem – ideal, natural e de topo plano (*flat-top*) – conforme mostra a Figura 7.30.

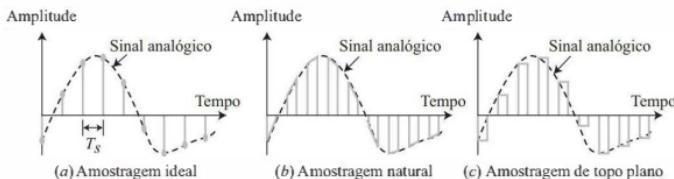



Figura 7.30 Três métodos diferentes de amostragem na PCM.

Na amostragem ideal, são amostrados pulsos do sinal analógico. É um método de amostragem ideal e não pode ser facilmente implementado. Na amostragem natural, um amostrador de alta velocidade é ativado apenas pelo pequeno intervalo de tempo no qual a amostragem ocorre. O resultado é uma sequência de amostras que retém a forma do sinal analógico. O método de amostragem mais comum, denominado **amostra e retém** (***sample and hold***), entretanto, cria amostras com topo plano (*flat-top*) usando um circuito.

O processo de amostragem é também denominado **Modulação de Amplitude de Pulso** (PAM – Pulse Amplitude Modulation). Precisamos nos lembrar, entretanto, que o resultado ainda é um sinal analógico com valores não inteiros.

Taxa de amostragem Uma consideração importante refere-se à taxa de amostragem ou frequência. Quais são as restrições sobre T_s? Essa pergunta foi elegantemente respondida por Nyquist. De acordo com o **teorema de Nyquist**, para reproduzir o sinal analógico original, uma condição necessária é que a **taxa de amostragem** seja pelo menos o dobro da frequência mais elevada do sinal original.

Precisamos discutir melhor o teorema neste ponto. Em primeiro lugar, podemos amostrar um sinal apenas se ele tiver uma banda limitada. Em outras palavras, um sinal com uma largura de banda infinita não pode ser amostrado. Em segundo lugar, o valor da taxa de amostragem deve ser pelo menos duas vezes o valor da frequência mais elevada, não da largura de banda. Se o sinal analógico for passa-baixas, a largura de banda e a frequência mais alta têm o mesmo valor. Se o sinal analógico for passa-faixa, o valor da largura de banda é menor do que o valor da frequência máxima. A Figura 7.31 mostra o valor da taxa de amostragem para esses dois tipos de sinal.

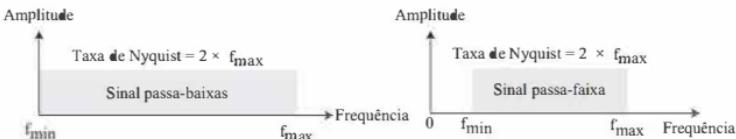



Figura 7.31 Taxa de amostragem de Nyquist para sinais passa-baixa e passa-faixa.

Quantização

O resultado da amostragem consiste em uma série de pulsos cujos valores de amplitude ficam entre as amplitudes máxima e mínima do sinal. O conjunto de amplitudes pode ser infinito, com valores não inteiros entre os dois limites. Esses valores não podem ser usados no processo de codificação. As etapas da quantização (também conhecida como quantificação) são as seguintes:

1. Consideramos que o sinal analógico original tem valores entre V_{min} e V_{max}.
2. Dividimos a faixa de valores em L zonas, cada uma apresentando altura Δ (delta).

$$\Delta = (V_{max} - V_{min}) / L$$

3. Atribuímos valores quantizados entre 0 e $L - 1$ para o ponto médio de cada zona.
4. O valor da amplitude amostrada é aproximado para o valor quantizado.

Como um exemplo simples, suponha que temos um sinal amostrado e as amplitudes da amostra estejam entre −20 e +20 V. Decidimos usar oito níveis ($L = 8$). Isto significa que o valor de Δ = 5 V. A Figura 7.32 mostra esse exemplo.

Apresentamos apenas nove amostras usando amostragem ideal (para simplificar). O valor no topo de cada amostra no gráfico mostra a amplitude real. No gráfico, a primeira linha corresponde ao valor normalizado para cada amostra (amplitude real/Δ). O processo de quantização seleciona o valor de quantização a partir do meio de cada zona. Isso significa que os valores normalizados quantizados (segunda linha) são diferentes das amplitudes normalizadas.

A diferença é denominada *erro normalizado* (terceira linha). A quarta linha é o código de quantização para cada amostra com base nos níveis de quantização na parte esquerda do gráfico. As palavras codificadas (quinta linha) são os produtos finais da conversão.

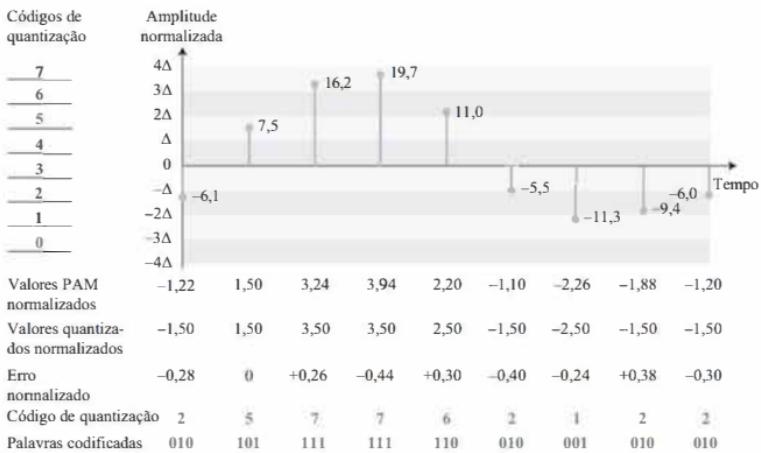



Figura 7.32 Quantização e codificação de um sinal amostrado.

Níveis de quantização No exemplo anterior, mostramos oito níveis de quantização. A escolha de L, o número de níveis, depende da faixa de amplitudes do sinal analógico e de como desejamos recuperar o sinal. Se a amplitude de um sinal varia apenas entre dois valores, precisamos de apenas dois níveis; se o sinal, tal como um sinal de voz, apresentar diversos valores de amplitude, precisamos de mais níveis de quantização. Na digitalização de áudio, L é normalmente escolhido como sendo 256; no caso de vídeo, L normalmente vale alguns milhares. A escolha de valores pequenos de L aumenta o erro de quantização se houver uma grande quantidade de variação no sinal.

Erro de quantização Uma questão importante refere-se ao erro criado no processo de quantização. A quantização é um processo de aproximação. Os valores de entrada do quantizador são os valores reais do sinal; os valores de saída são os valores aproximados. Cada valor de saída é escolhido como o valor no meio da zona. Se o valor de entrada também estiver no meio da zona, não há qualquer erro de quantização; caso contrário, existe um erro. No exemplo anterior, a amplitude normalizada da terceira amostra é 3,24, mas o valor normalizado quantizado é 3,50. Isto significa que há um erro de +0,26. O valor do erro para qualquer amostra é inferior a $\Delta/2$. Em outras palavras, temos $-/2 \leq \text{erro} \leq \Delta/2$.

O erro de quantização altera a relação sinal/ruído do sinal, o que por sua vez reduz o limite superior da capacidade de acordo com Shannon.

Pode-se provar que a contribuição do *erro de quantização* na S/R_{dB} do sinal depende do número de níveis de quantização L, ou do número de *bits* por amostra n_b, conforme mostrado na seguinte fórmula.

$$S/R_{dB} = 6{,}02 n_b = 1{,}76 \text{ dB}$$

Codificação

A última etapa da PCM é a codificação. Depois de cada amostra ter sido quantizada e do número de *bits* por amostra ter sido decidido, cada amostra pode ser transformada em uma palavra

de código de n_b *bits*. Na Figura 7.32, as palavras codificadas são mostradas na última linha. Um código de quantização de valor 2 é codificado como 010; 5 é codificado como 101, e assim por diante. Perceba que o número de *bits* de cada amostra é determinado a partir do número de níveis de quantização. Se o número de níveis de quantização for L, o número de *bits* é $n_b = \log_2 L$. No nosso exemplo, L vale 8 e n_b é, portanto, 3. A taxa de *bits* pode ser determinada usando a fórmula a seguir.

Taxa de *bits* = taxa de amostragem × número de *bits* por amostra = $f_s \times n_b$

Exemplo 7.13

Queremos digitalizar a voz humana. Qual é a taxa de *bits* necessária, considerando 8 *bits* por amostra?

Solução

A voz humana normalmente contém frequências entre 0 e 4000 Hz. Portanto, a taxa de amostragem e a taxa de *bits* são calculadas conforme se segue.

Taxa de amostragem = 4.000 × 2 = 8.000 amostras/s → Taxa de bits = 8.000 × 8 = 64.000 bps = 64 kbps

Recuperação do sinal original

A recuperação do sinal original requer o decodificador PCM, que primeiramente utiliza mecanismos para converter as palavras de código em um pulso que mantém sua amplitude até o pulso seguinte. Após o sinal de degrau ter sido completado, ele passa por um filtro passa-baixas para suavizar o degrau, transformando o sinal em analógico. O filtro tem a mesma frequência de corte que o sinal original no emissor. Se o sinal for amostrado usando a taxa de amostragem de Nyquist (ou uma taxa superior) e se houver níveis suficientes de quantização, o sinal original será recuperado. Perceba que os valores máximo e mínimo do sinal original podem ser obtidos amplificando-se o sinal. A Figura 7.33 mostra o processo simplificado.

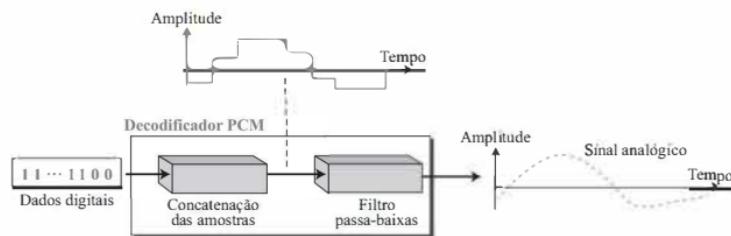



Figura 7.33 Componentes de um decodificador PCM.

Largura de banda PCM

Pode-se provar que a largura de banda mínima do sinal digital é aquela mostrada a seguir.

$$B_{min} = n_b \times B_{analógica}$$

Isto significa que a largura de banda mínima do sinal digital é n_b vezes maior do que a largura de banda do sinal analógico. Esse é o preço que pagamos pela digitalização.

Modulação Delta

A PCM é uma técnica muito complexa. Outras técnicas foram desenvolvidas para reduzir a complexidade da PCM. A mais simples delas é a **Modulação Delta** (DM – Delta Modulation). A PCM é usada para determinar o valor da amplitude do sinal para cada amostra; já a DM determina a variação com relação à amostra anterior. A Figura 7.34 mostra o processo. Perceba que não há palavras de código aqui; os *bits* são enviados um após o outro.

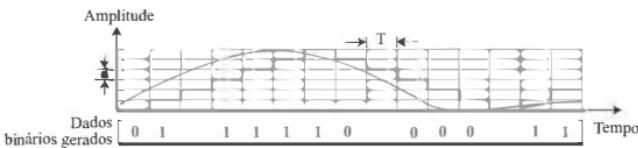



Figura 7.34 O processo de Modulação Delta.

7.3 TRANSMISSÃO ANALÓGICA

Apesar de a transmissão digital ser desejável, ela requer um canal passa-baixas; a transmissão analógica é a única opção se tivermos um canal passa-faixa. O ato de converter dados digitais em um sinal analógico passa-faixa é tradicionalmente denominado *conversão digital-analógico*. O ato de converter um sinal analógico passa-baixas em um sinal analógico passa-faixa é tradicionalmente denominado *conversão analógico-analógico*. Nesta seção, discutimos esses dois tipos de conversão.

7.3.1 Conversão digital-analógico

A **conversão digital-analógico** consiste no processo de transformar uma das características de um sinal analógico com base na informação dos dados digitais. A Figura 7.35 mostra a relação entre a informação digital, o processo de modulação digital-analógico e o sinal analógico resultante.

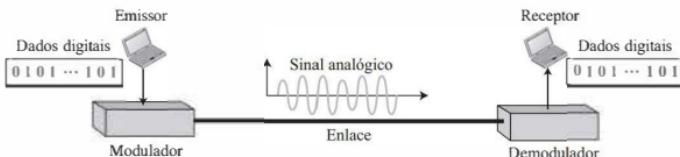



Figura 7.35 Conversão digital-analógico.

Conforme discutido anteriormente, uma onda senoidal é definida por três características: amplitude, frequência e fase. Quando variamos qualquer uma dessas características, criamos uma versão diferente dessa onda. Assim, variando uma característica de um sinal elétrico simples, podemos usá-lo para representar dados digitais. Qualquer uma das três características seguintes pode ser alterada dessa forma, o que nos dá pelo menos três mecanismos para modular

dados digitais em um sinal analógico: **Modulação por Chaveamento de Amplitude** (ASK – Amplitude Shift Keying), **Modulação por Chaveamento de Frequência** (FSK – Frequency Shift Keying) e **Modulação por Chaveamento de Fase** (PSK – Phase Shift Keying). Além disso, existe um quarto (e melhor) mecanismo que combina a modificação de amplitude e de fase, denominado **Modulação por Amplitude em Quadratura** (QAM – Quadrature Amplitude Modulation). A modulação QAM é a mais eficiente dentre essas opções e é o mecanismo geralmente utilizado atualmente.

Modulação por chaveamento de amplitude

Na modulação por chaveamento de amplitude (ASK), a amplitude do sinal de portadora é variada para criar elementos de sinal. Frequência e fase permanecem constantes enquanto a amplitude varia.

ASK binária

A modulação ASK é normalmente implementada usando apenas dois níveis. Essa estratégia é conhecida como *modulação ASK binária* (BASK) ou como *chaveamento liga-desliga* (OOK – On-Off Keying). A amplitude de pico de um dos níveis do sinal é 0; a outra é igual à amplitude do sinal de portadora. A Figura 7.36 dá uma visão conceitual do ASK binário. A Figura 7.36 também mostra a largura de banda usada pela modulação ASK. Embora o sinal de portadora seja apenas uma onda senoidal simples, o processo de modulação produz um sinal composto aperiódico. Este, conforme discutimos anteriormente, apresenta um conjunto contínuo de frequências. Como esperado, a largura de banda é proporcional à taxa de transferência do sinal (*baud*). Entretanto, existe normalmente outro fator envolvido, denotado d, que depende da modulação e do processo de filtragem. O valor de d fica entre 0 e 1. Isto significa que a largura de banda pode ser expressa conforme mostrado na figura, onde S é a taxa de transferência do sinal e B é a largura de banda.

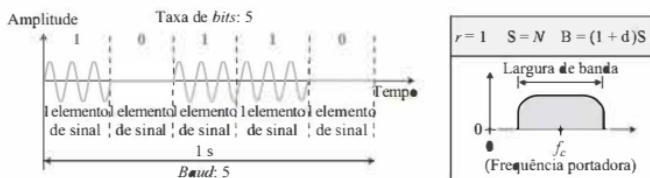



Figura 7.36 Modulação ASK binária (BASK).

A fórmula, $B = (1 + d)\, S$, mostra que a largura de banda necessária tem um valor mínimo de S e um valor máximo de $2S$. O ponto mais importante aqui é a localização da largura de banda. O meio da largura de banda é onde f_c, a frequência portadora, está localizada. Isto significa que se tivermos um canal passa-faixa disponível, podemos escolher nosso valor de f_c de modo que o sinal modulado ocupe aquela largura de banda. Essa é, de fato, a vantagem mais significativa da conversão digital-analógico. Podemos deslocar a largura de banda resultante para corresponder àquela que se encontra disponível.

ASK multinível

A discussão anterior utiliza apenas dois níveis de amplitude. Podemos ter um ASK multinível, no qual existem mais do que dois níveis. Podemos usar 4, 8, 16 ou um número maior de amplitudes diferentes para o sinal e modular os dados usando 2, 3, 4 ou mais *bits* de cada vez. Nesses casos, $r = 2$, $r = 3$, $r = 4$ e assim por diante. Embora isto não seja implementado noASK puro, é implementado no QAM (conforme veremos mais adiante).

Modulação por chaveamento de frequência

Na modulação por chaveamento de frequência (FSK), a frequência do sinal de portadora varia para representar os dados. A frequência do sinal modulado permanece constante por toda a duração de um elemento de sinal, mas se altera para o elemento de sinal seguinte se o elemento de dados mudar. A amplitude de pico e a fase permanecem constantes para todos os elementos de sinal.

FSK binária

Uma maneira de enxergar a FSK binária (ou BFSK) é considerar duas frequências portadoras. Na Figura 7.37, escolhemos duas frequências portadoras, f_1 e f_2. Usamos a primeira portadora se o elemento de dados for 0; usamos a segunda se o elemento de dados for 1. Entretanto, perceba que esse é um exemplo não realista, usado apenas para fins de demonstração. Normalmente, as frequências portadoras são muito elevadas e a diferença entre elas é muito pequena.

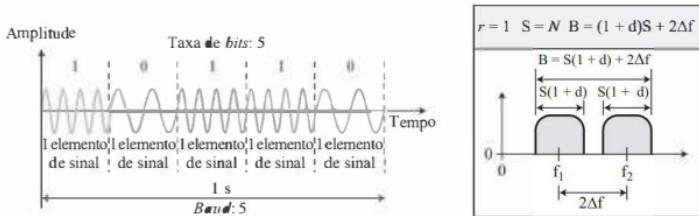



Figura 7.37 FSK binária.

Conforme mostra a Figura 7.37, o meio de uma largura de banda é f_1 e o meio da outra é f_2. Tanto f_1 como f_2 são separados por uma distância Δ_f do ponto médio entre as duas bandas. A diferença entre as duas frequências é $2\Delta_f$.

A Figura 7.37 mostra também a largura de banda usada pela modulação BFSK. Novamente, os sinais de portadora são apenas ondas senoidais simples, porém a modulação cria um sinal aperiódico composto por frequências contínuas. Podemos enxergar o FSK como dois sinais ASK, cada um com sua própria frequência portadora (f_1 ou f_2). Se a diferença entre as duas frequências for $2\Delta_f$, então a largura de banda necessária é $\boldsymbol{B = (1 + d) \times S + 2\Delta_f}$.

Qual deve ser o valor mínimo de $2\Delta_f$? Na Figura 7.37, escolhemos um valor maior do que $(1 + d)\,S$. Pode-se demonstrar que o valor mínimo deve ser pelo menos S para o bom funcionamento dos processos de modulação e demodulação.

FSK multinível

Não é incomum aplicar modulação multinível com o método FSK (o resultado é conhecido como MFSK). Podemos usar mais do que duas frequências. Por exemplo, podemos usar quatro frequências diferentes f_1, f_2, f_3 e f_4 para enviar 2 *bits* de cada vez. Para enviar 3 *bits* de cada vez, podemos usar oito frequências. E assim por diante. No entanto, precisamos lembrar que as frequências devem estar separadas por uma distância de $2\Delta_f$. Para que o modulador e o demodulador operem corretamente, pode-se demonstrar que o valor mínimo de $2\Delta_f$ precisa ser S. Podemos mostrar que a largura de banda para $d = 0$ é

$$\boldsymbol{B = (1 + d) \times S + (L - 1)\, 2\Delta_f \quad \rightarrow \quad B = L \times S.}$$

Modulação por chaveamento de fase

Na modulação por chaveamento de fase (PSK), a fase da portadora varia para representar dois ou mais elementos de sinal diferentes. A amplitude de pico e a frequência permanecem ambas constantes enquanto a fase muda. Atualmente, o PSK é mais usado que o ASK ou o FSK. No entanto, veremos em breve que o QAM, que combina ASK e PSK, é o método dominante de modulação digital-analógico.

PSK binário

O PSK mais simples é o PSK binário (BPSK), no qual temos apenas dois elementos de sinal, um com uma fase de 0° e o outro com uma fase de 180°. A Figura 7.38 fornece uma visão conceitual do PSK. O PSK binário é tão simples quanto o ASK binário, mas com uma grande vantagem: ele é menos suscetível a ruídos. No ASK, o critério para a detecção de *bits* é a amplitude do sinal; no PSK, a fase é usada para esse propósito. O ruído é capaz de alterar a amplitude mais facilmente do que a fase. O PSK é superior ao FSK porque não precisamos de dois sinais de portadora.

A Figura 7.38 também mostra a largura de banda usada pela modulação BPSK, igual àquela usada pelo ASK binário, mas é menor que a usada pelo BFSK. Não há desperdício de largura de banda para separar dois sinais de portadora.

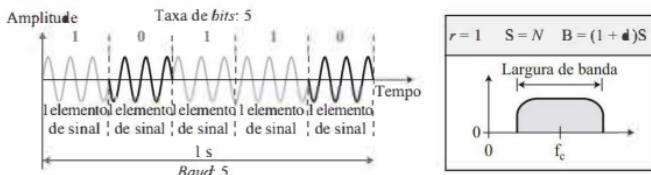



Figura 7.38 PSK binária.

PSK em quadratura

A simplicidade do BPSK incentivou projetistas a usar 2 *bits* de cada vez em cada elemento de sinal, diminuindo, assim, a taxa de *baud* e, consequentemente, a largura de banda necessária. O esquema resultante é denominado *PSK em Quadratura* ou *QPSK*, porque ele usa duas modulações BPSK separadas; uma delas fica em fase, a outra em quadratura (fora de fase por 90 graus). Os *bits* de entrada passam primeiro por um conversor de série para paralelo, que envia um *bit* para um modulador e o próximo *bit* para o outro modulador. Se a duração de cada *bit* no sinal de entrada for *T*, a duração de cada *bit* enviado pelo sinal BPSK correspondente será *2T*. Isto significa que o *bit* em cada sinal BPSK tem metade da frequência que tinha no sinal original.

Diagrama de constelação

Um **diagrama de constelação** pode nos ajudar a definir a amplitude e a fase de um elemento de sinal, particularmente quando utilizamos duas portadoras (uma em fase e outra em quadratura). O diagrama é útil quando estamos lidando com modulações multinível, sejam elas ASK, PSK ou QAM (ver próxima seção). Em um diagrama de constelação, um elemento de sinal é representado como um ponto. O *bit* ou a combinação de *bits* que ele transporta é geralmente escrito ao lado dele.

O diagrama tem dois eixos. O eixo horizontal *X* refere-se à portadora em fase; o eixo vertical *Y* refere-se à portadora em quadratura. Para cada ponto no diagrama, quatro informações podem ser deduzidas. A projeção do ponto sobre o eixo *X* define a amplitude de pico do componente em fase; a projeção do ponto sobre o eixo *Y* define a amplitude de pico do componente em quadratura. O comprimento da linha (vetor) que liga o ponto à origem fornece a amplitude de pico do elemento

de sinal (a combinação dos componentes *X* e *Y*); o ângulo que a linha faz com o eixo *X* corresponde à fase do elemento de sinal. Portanto, todas as informações necessárias podem ser facilmente encontradas em um diagrama de constelação. A Figura 7.39 ilustra a ideia com três diagramas de constelação, um para cada sinal BASK (OOK), BPSK e QPSK.

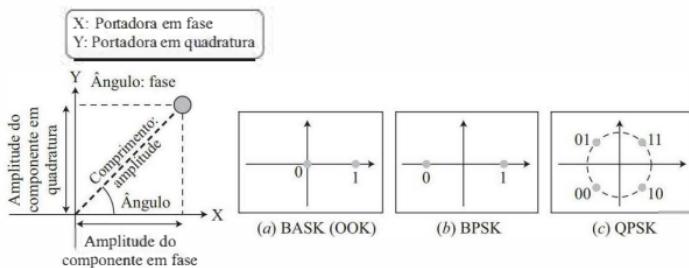



Figura 7.39 Conceito de um diagrama de constelação.

Modulação por Amplitude em Quadratura

O PSK é limitado pela capacidade do equipamento em distinguir pequenas diferenças de fase. Esse fator limita sua taxa de *bits* em potencial. Até aqui, mostramos esquemas que alteram apenas uma das três características de uma onda senoidal por vez; mas e se alterarmos duas? Por que não combinamos o ASK e o PSK? A ideia de utilizar duas portadoras, uma em fase e outra em quadratura, com níveis de amplitude diferentes para cada portadora, é o conceito subjacente à **Modulação por Amplitude em Quadratura** (Modulação QAM – Quadrature Amplitude Modulation).

Existem numerosas variações possíveis para a modulação QAM. A Figura 7.40 mostra alguns desses esquemas. A Figura 7.40*a* mostra o esquema mais simples, o 4-QAM (quatro tipos diferentes de elementos de sinal), usando um sinal NRZ unipolar para modular cada portadora. Esse é o mesmo mecanismo que usamos no BASK (OOK). A parte *b* da figura mostra outro esquema 4-QAM usando NRZ polar, mas o resultado é exatamente igual ao do esquema QPSK. A parte *c* mostra outro esquema 4-QAM no qual foi utilizado um sinal com dois níveis positivos para modular cada uma das duas portadoras. Finalmente, a Figura 7.40*d* mostra uma constelação 16-QAM de um sinal com oito níveis, quatro positivos e quatro negativos.

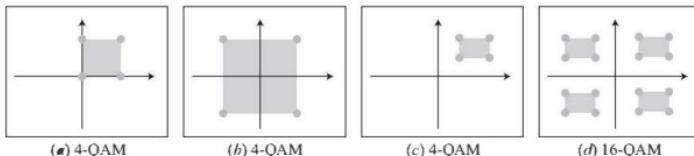



Figura 7.40 Diagramas de constelação para alguns QAMs.

Largura de banda do QAM

A largura de banda mínima necessária para a transmissão usando QAM é igual àquela exigida pela modulação ASK e pela modulação PSK. A modulação QAM apresenta as mesmas vantagens que a modulação PSK apresenta sobre a modulação ASK.

7.3.2 Conversão analógico-analógico

A conversão analógico-analógico, ou modulação analógica, consiste na representação da informação analógica por meio de um sinal analógico. Podemos nos perguntar por que precisamos modular um sinal analógico; afinal, ele já é analógico. A modulação é necessária se o meio for naturalmente passa-faixa ou se tivermos à nossa disposição apenas um canal passa-faixa. Um exemplo é o rádio. O governo atribui uma largura de banda estreita para cada estação de rádio. O sinal analógico produzido por cada estação é um sinal passa-baixas, e todos eles ocupam a mesma faixa. Para conseguirmos ouvir estações diferentes, os sinais passa-baixas precisam ser deslocados, cada um para uma posição diferente.

A **conversão analógico-analógico** pode ser realizada de três maneiras: **Modulação de Amplitude** (AM – Amplitude Modulation), **Modulação de Frequência** (FM – Frequency Modulation) e **Modulação de Fase** (PM – Phase Modulation). As modulações FM e PM são geralmente classificadas como uma só.

Modulação de Amplitude (AM)

Na transmissão AM, o sinal de portadora é modulado de forma que a sua amplitude se altera com as variações de amplitude do sinal que está sendo modulado. A frequência e a fase da portadora permanecem inalteradas; apenas a amplitude se altera de acordo com as variações da informação. A Figura 7.41 mostra como esse conceito funciona. O sinal que está sendo modulado é o envelope da portadora.

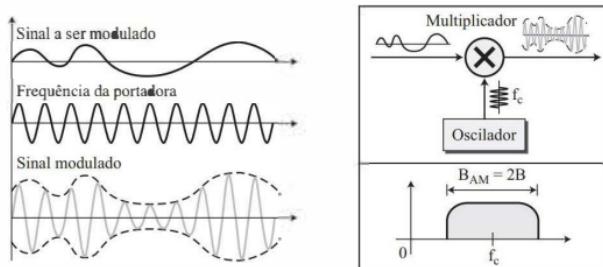



Figura 7.41 Modulação de amplitude.

Conforme mostra a Figura 7.41, a modulação AM é normalmente implementada usando um multiplicador simples, porque a amplitude do sinal de portadora precisa ser modificada de acordo com a amplitude do sinal que está sendo modulado. A Figura 7.41 mostra também a largura de banda de um sinal AM. A modulação cria uma largura de banda que é o dobro da largura de banda do sinal que está sendo modulado e cobre uma faixa centrada na frequência portadora. Entretanto, os componentes de sinal acima e abaixo da frequência portadora transportam exatamente a mesma informação. Por isso, algumas implementações descartam metade dos sinais e cortam a largura de banda ao meio.

Modulação de Frequência (FM)

Na transmissão FM, a frequência do sinal de portadora é modulada de forma a seguir as mudanças no nível de tensão (amplitude) do sinal que está sendo modulado. A amplitude de pico

e a fase do sinal de portadora permanecem constantes, mas à medida que a amplitude do sinal que carrega a informação se altera, a frequência da portadora muda de forma correspondente. A Figura 7.42 mostra as relações entre o sinal que está sendo modulado, o sinal da portadora e o sinal FM resultante.

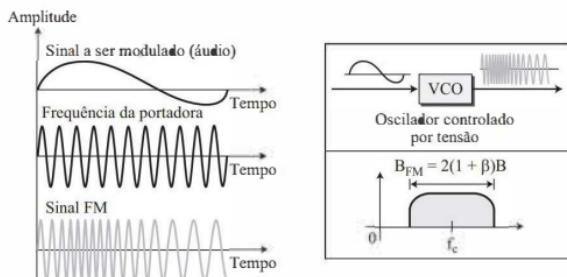



Figura 7.42 Modulação de frequência.

Conforme mostra a Figura 7.42, a modulação FM é normalmente implementada usando um oscilador controlado por tensão (voltagem), como ocorre no FSK. A frequência do oscilador muda de acordo com a tensão de entrada, que é a amplitude do sinal que está sendo modulado. A Figura 7.42 mostra também a largura de banda de um sinal FM. É difícil determinar a largura de banda efetiva com exatidão, mas pode ser calculada empiricamente como $B_{FM} = 2(1 + \beta)B$, onde β é um fator que depende da técnica de modulação, mas cujo valor geralmente é 4.

Modulação de Fase (PM)

Na transmissão PM, a fase do sinal de portadora é modulada para variar conforme a variação do nível de tensão (amplitude) do sinal sendo modulado. A amplitude de pico e a frequência do sinal de portadora permanecem constantes, mas à medida que a amplitude do sinal que carrega a informação se altera, a fase da portadora muda de forma correspondente. Pode-se demonstrar matematicamente que a modulação PM é igual à FM com uma diferença. Na modulação FM, a variação instantânea na frequência portadora é proporcional à amplitude do sinal que está sendo modulado; na modulação PM, a variação instantânea na frequência portadora é proporcional à derivada da amplitude do sinal que está sendo modulado. A Figura 7.43 mostra as relações entre o sinal que está sendo modulado, o sinal da portadora e o sinal PM resultante.

Conforme mostra a Figura 7.43, a modulação PM é normalmente implementada usando um oscilador controlado por tensão acoplado a um medidor de derivada. A frequência do oscilador varia de acordo com a derivada da tensão de entrada, que corresponde à amplitude do sinal que está sendo modulado.

A Figura 7.43 também mostra a largura de banda de um sinal AM. É difícil determinar a largura de banda efetiva com exatidão, mas pode-se mostrar empiricamente que seu valor é várias vezes a largura do sinal analógico. Embora a fórmula mostre a mesma largura de banda para a modulação FM e para a modulação PM, o valor de β é menor no caso da modulação PM (cerca de 1 para uma banda estreita e de 3 para uma banda larga).

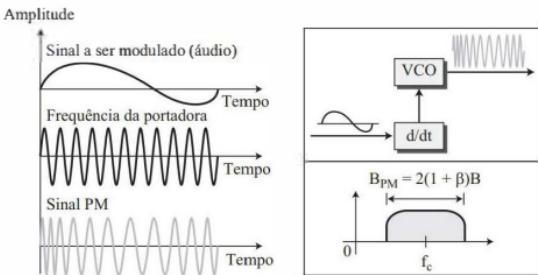



Figura 7.43 Modulação de fase.

7.4 UTILIZAÇÃO DE BANDA

Na vida real, encontramos conexões físicas com largura de banda limitada. O uso criterioso dessas bandas tem sido, e continuará a ser, um dos principais desafios das comunicações eletrônicas. Entretanto, o significado de *criterioso* pode depender da aplicação. Em alguns casos, precisamos combinar vários canais que tenham uma largura de banda reduzida para fazer melhor uso de um canal apresentando maior largura de banda. Em outros casos, precisamos expandir a largura de banda de um canal para atingir objetivos como privacidade e resistência a interferências. Nesta seção, exploramos essas duas grandes categorias de utilização de banda: *multiplexação* e *espalhamento*. Na multiplexação, nosso objetivo é aumentar a eficiência do canal; combinamos diversos canais em um só. No espalhamento, nosso objetivo é aumentar a privacidade e a resistência a interferências; expandimos a largura de banda de um canal para inserir redundância, algo necessário para atingir esse objetivo.

7.4.1 Multiplexação

Sempre que a largura de banda de um meio ligando dois dispositivos é maior do que a largura de banda exigida pelos dispositivos, o meio pode ser compartilhado. A **multiplexação** consiste no conjunto de técnicas que permite a transmissão simultânea de múltiplos sinais através de um único enlace de dados. À medida que aumenta o uso de dados e dos meios de telecomunicação, aumenta também o tráfego. Podemos acomodar esse aumento se continuarmos a criar enlaces individuais toda vez que um novo canal passar a ser necessário, ou podemos instalar enlaces de maior largura de banda e usar cada um para transportar múltiplos sinais.Tecnologias atuais fornecem meios com elevada largura de banda, como fibra óptica e micro-ondas terrestres e via satélite. Cada uma dessas tecnologias apresenta uma largura de banda muito superior àquela necessária para a transmissão de sinais comuns. Se um enlace fornece uma largura de banda maior do que aquela necessária pelos dispositivos ligados a ele, a largura de banda acaba desperdiçada. Um sistema eficiente é capaz de maximizar a utilização de todos os recursos; a largura de banda é um dos recursos mais preciosos existentes no contexto de comunicação de dados.

Em um sistema multiplexado, *n* linhas compartilham a largura de banda de um enlace. A Figura 7.44 mostra o formato básico de um sistema multiplexado. As linhas na esquerda direcionam seus fluxos de transmissão para um *multiplexador*, que os combina em um único fluxo (muitos-para-um). No lado do receptor, o fluxo passa por um *demultiplexador*, que separa novamente o fluxo em suas componentes de transmissão (um-para-muitos) e as direciona para as linhas de saída correspondentes. Na figura, a palavra *enlace* se refere ao caminho físico. A palavra *canal* se refere à porção de um enlace que carrega uma transmissão entre um certo par de linhas. Um enlace pode ter muitos (*n*) canais.

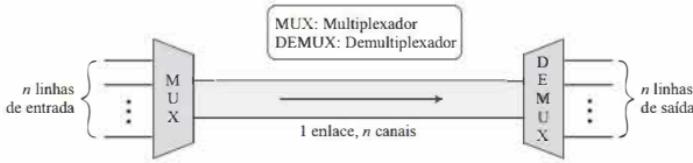



Figura 7.44 Dividindo um enlace em canais.

Existem três técnicas básicas de multiplexação: Multiplexação por Divisão de Frequência (FDM – Frequency-Division Multiplexing), Multiplexação por Divisão de Comprimento de Onda (WDM – Wavelength-Division Multiplexing) e Multiplexação por Divisão de Tempo (TDM – Time-Division Multiplexing). As duas primeiras são técnicas projetadas para sinais analógicos, e a terceira para sinais digitais.

Alguns livros usam os termos FDMA e FDM (ou TDMA e TDM) como sendo equivalentes. Acreditamos que FDMA e TDMA são protocolos de acesso definidos na camada de enlace de dados e que usam os serviços da FDM e da TDM na camada física. Discutimos o FDMA e o TDMA no Capítulo 6. Além disso, alguns livros tratam o Acesso Múltiplo por Divisão de Código (CDMA – Code Division Multiple Access) como uma quarta categoria de multiplexação. Discutimos o CDMA como um método de acesso na camada de enlace de dados também no Capítulo 6; não há um método correspondente na camada física.

Multiplexação por Divisão de Frequência

A **Multiplexação por Divisão de Frequência** (FDM – Frequency-Division Multiplexing) é uma técnica analógica que pode ser aplicada quando a largura de banda de um enlace (em hertz) for maior do que as larguras de banda combinadas dos sinais a serem transmitidos. Na FDM, os sinais gerados por cada dispositivo emissor modulam frequências portadoras diferentes. Esses sinais modulados são, então, combinados em um único sinal composto que pode ser transportado pelo meio físico. Frequências portadoras são separadas por uma largura de banda grande o suficiente para acomodar o sinal modulado. Essas faixas de largura de banda podem ser vistas como canais através dos quais os vários sinais viajam. Os canais podem ser separados por faixas de banda não utilizadas – **bandas de guarda** – para evitar que os sinais se sobreponham. Além disso, frequências portadoras não devem interferir com as frequências de dados originais.

A Figura 7.45 fornece uma visão conceitual da FDM. Nela, o caminho de transmissão é dividido em três partes, cada uma representando um canal que comporta uma transmissão.

Dissemos que a FDM é uma técnica de multiplexação analógica; entretanto, isto não significa que ela não pode ser usada para combinar dados de fontes emitindo sinais digitais. Estes podem ser convertidos em sinais analógicos antes que a FDM seja aplicada para multiplexá-los.

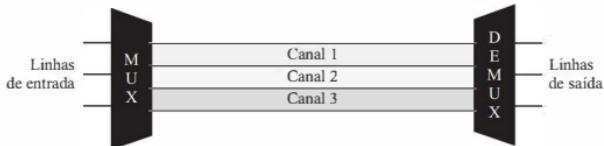



Figura 7.45 Multiplexação por divisão de frequência.

Exemplo 7.14

Considere que um canal de voz ocupa uma largura de banda de 4 kHz. Precisamos combinar três canais de voz em um enlace com uma largura de banda de 12 kHz, indo de 20 até 32 kHz. Mostre a configuração correspondente a esse cenário, usando o domínio da frequência. Considere que não existem bandas de guarda.

Solução

Deslocamos (modulamos) cada um dos três canais de voz para uma banda diferente, conforme mostrado na Figura 7.46. Usamos a banda de 20 a 24 kHz para o primeiro canal, a banda de 24 a 28 kHz para o segundo e a banda de 28 a 32 kHz para o terceiro. Em seguida, combinamos esses canais, conforme mostrado na Figura 7.46. No receptor, cada canal recebe o sinal completo e então utiliza um filtro para recuperar o seu próprio sinal. O primeiro canal usa um filtro que permite a passagem de frequências entre 20 e 24 kHz e filtra (descarta) quaisquer outras frequências. O segundo canal utiliza um filtro que permite a passagem de frequências entre 24 e 28 kHz e o terceiro canal usa um filtro que permite a passagem de frequências entre 28 e 32 kHz. Finalmente, cada canal desloca a frequência de modo que ela se inicie em zero.

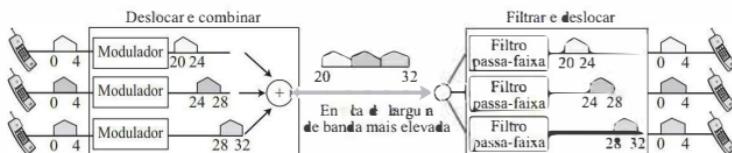



Figura 7.46 Esquema para o Exemplo 7.14.

Multiplexação por Divisão de Comprimento de Onda

A técnica de **Multiplexação por Divisão de Comprimento de Onda** (WDM – Wavelength-Division Multiplexing) foi projetada para usar a elevada capacidade de transferência de dados dos cabos de fibra óptica. A taxa de transferência de dados da fibra óptica é maior do que a taxa de transferência de dados de um cabo de transmissão metálico. O uso de um cabo de fibra óptica para uma única linha acaba desperdiçando a largura de banda disponível. A multiplexação nos permite combinar os sinais de diversas linhas em um só.

A técnica de WDM é conceitualmente equivalente à de FDM, exceto que a multiplexação e a demultiplexação envolvem sinais ópticos transmitidos por canais de fibra óptica. A ideia é a mesma: estamos combinando sinais distintos de frequências distintas. A diferença é que as frequências são muito elevadas.

A Figura 7.47 fornece uma visão conceitual de um multiplexador e de um demultiplexador WDM. Bandas muito estreitas de luz provenientes de diferentes fontes são combinadas para criar uma ampla banda de luz. No receptor, os sinais são separados pelo demultiplexador.

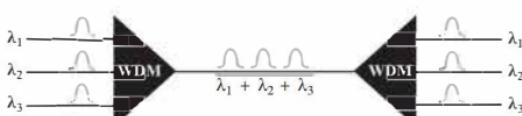



Figura 7.47 Multiplexação por Divisão de Comprimento de Onda.

Uma aplicação da técnica de WDM é a rede SONET, na qual múltiplas linhas de fibra óptica são multiplexadas e demultiplexadas. Discutimos SONET no Capítulo 5.

Multiplexação por Divisão de Tempo

A técnica de **Multiplexação por Divisão de Tempo** (TDM – Time-Division Multiplexing) é um processo digital que permite que várias conexões compartilhem a largura de banda de um enlace. Em vez de compartilhar uma parte da largura de banda como ocorre na técnica de FDM, o tempo é compartilhado. Cada conexão ocupa uma parcela de tempo no enlace. A Figura 7.48 fornece uma visão conceitual da TDM. Perceba que o mesmo enlace é utilizado, assim como na FDM; nesse caso, entretanto, o enlace mostrado é dividido em seções por tempo e não por frequência. Na figura, as porções dos sinais 1, 2, 3 e 4 ocupam o enlace sequencialmente.

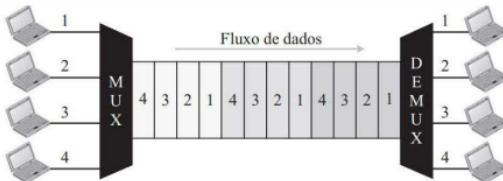



Figura 7.48 TDM

Perceba que, na Figura 7.48, estamos preocupados apenas com a multiplexação, não com a comutação. Isto significa que todos os dados de uma mensagem provenientes da fonte 1 sempre vão para um destino específico, seja ela 1, 2, 3 ou 4. A entrega é fixa e invariável, ao contrário do que ocorre na comutação.

Precisamos também lembrar que a técnica de TDM é, em princípio, uma técnica de multiplexação digital. Os dados digitais provenientes de diferentes fontes são combinados em um enlace compartilhado no tempo. No entanto, isto não significa que as fontes não podem gerar dados analógicos; estes podem ser amostrados, transformados em dados digitais e, em seguida, multiplexados usando TDM.

Podemos dividir a técnica de TDM em dois esquemas diferentes: síncrono e estatístico. Discutimos primeiramente a **TDM síncrona** e, em seguida, mostramos quais as diferenças da **TDM estatística**. Na TDM síncrona, cada conexão de entrada tem uma saída a ela alocada, mesmo que não esteja enviando dados.

TDM síncrona

Na TDM síncrona, o fluxo de dados de cada conexão de entrada é dividido em unidades, de modo que cada entrada ocupa uma parcela de tempo na entrada. Uma unidade pode ser um 1 *bit*, um caractere ou um bloco de dados. Cada unidade de entrada torna-se uma unidade de saída e ocupa uma parcela de tempo de saída, porém a duração de uma parcela de tempo de saída é n vezes menor do que a duração de uma parcela de tempo de entrada. Se uma parcela de tempo de entrada tiver T segundos, a parcela de tempo de saída terá T/n segundos, onde n é o número de conexões. Em outras palavras, uma unidade na conexão de saída tem uma duração mais curta, porém ela viaja mais rápido. A Figura 7.49 mostra um exemplo de TDM síncrona na qual n vale 3.

Na TDM síncrona, um conjunto de unidades de dados de cada conexão de entrada é agrupado em um quadro (veremos a razão para isso mais adiante). Se tivermos n conexões, cada quadro é dividido em n parcelas de tempo e uma parcela de tempo é alocada a cada unidade, uma de cada linha de entrada. Se a duração da unidade de entrada for T, a duração de cada parcela será T/n e a de cada quadro será T (a menos que um quadro carregue alguma outra informação, conforme veremos a seguir).

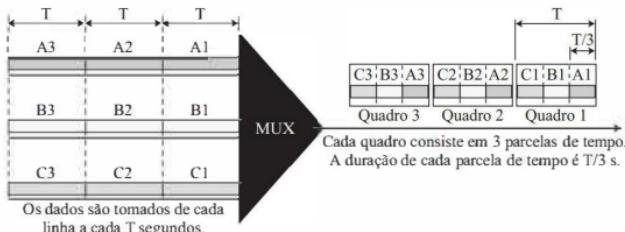



Figura 7.49 Multiplexação por Divisão de Tempo Síncrona.

A taxa de transferência de dados do enlace de saída deve ser *n* vezes a taxa de transferência de dados de uma conexão física para suportar o fluxo de dados. Na Figura 7.49, a taxa de transferência de dados do enlace é 3 vezes a taxa de transferência de dados de uma conexão; de modo similar, a duração de uma unidade em uma conexão equivale a 3 vezes o valor da parcela de tempo (duração de uma unidade no enlace). Na figura, representamos os dados antes da multiplexação com um tamanho igual a 3 vezes o tamanho dos dados após a multiplexação. Isto é feito apenas para transmitir a ideia de que a duração de cada unidade é 3 vezes maior antes da multiplexação do que depois dela.

As parcelas de tempo são agrupadas em quadros. Um quadro consiste em um ciclo completo de parcelas de tempo, com uma parcela dedicada a cada dispositivo emissor. Em um sistema com *n* linhas de entrada, cada quadro tem *n* parcelas de tempo, e cada parcela é alocada para transportar dados provenientes de uma linha de entrada específica.

Exemplo 7.15

A Figura 7.50 mostra a TDM síncrona com um fluxo de dados para cada entrada e um fluxo de dados para a saída. A unidade de dados é um *bit*. Determine (*a*) a duração dos *bits* de entrada, (*b*) a duração dos *bits* de saída, (*c*) a taxa de saída de *bit* e (*d*) a taxa saída de quadros.

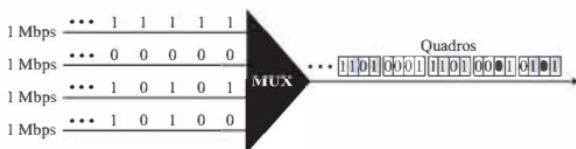



Figura 7.50 Esquema para o Exemplo 7.15.

Solução

Podemos responder às perguntas da seguinte forma:

1. A duração dos *bits* de entrada corresponde ao inverso da taxa de *bits*: 1/1 Mbps = 1 μs.
2. A duração dos *bits* de saída corresponde a um quarto da duração dos *bits* de entrada, ou 1/4 μs.
3. A taxa de saída de *bits* é o inverso da duração dos *bits* de saída, ou 1/(1/4) μs ou 4 Mbps. Esse valor também pode ser deduzido do fato de a taxa de saída ser 4 vezes o valor de qualquer taxa de entrada; portanto, a taxa de saída pode ser calculada como 4 × 1 Mbps = 4 Mbps.

4. A taxa de transferência de quadros é sempre igual a qualquer taxa de entrada. Assim, a taxa de quadros vale 1.000.000 quadros por segundo. Como estamos enviando 4 *bits* em cada quadro, podemos verificar o resultado da questão anterior multiplicando a taxa de quadros pelo número de *bits* em cada quadro.

Exemplo 7.16

As companhias telefônicas implementam a TDM por meio de uma hierarquia de sinais digitais denominada **serviço de sinal digital** (DS – digital signal) ou hierarquia digital. A Figura 7.51 mostra as taxas de transferência de dados suportadas por cada nível. As implementações comerciais desses serviços são conhecidas como **linhas T**.

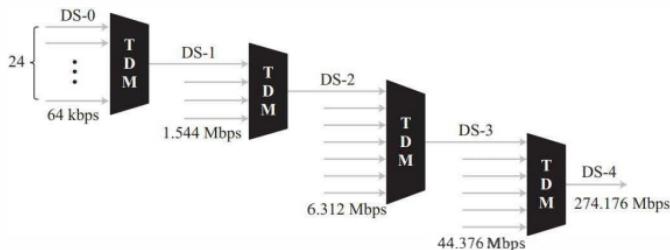



Figura 7.51 Hierarquia digital.

- O serviço **DS-0** corresponde a um único canal digital de 64 kbps.
- O **DS-1** é um serviço de 1.544 Mbps; 1.544 Mbps são 24 vezes 64 kbps mais 8 kbps de carga adicional (*overhead* causado pelo serviço). Esse serviço pode ser usado como um único serviço para transmissões a 1.544 Mbps, ou para multiplexar 24 canais DS-0, ou então para transportar qualquer outra combinação desejada pelo usuário que possa ser encaixada dentro de sua capacidade de 1.544 Mbps. A aplicação comercial desse serviço é conhecida como uma linha **T-1**.
- O **DS-2** é um serviço de 6.312-Mbps; 6.312 Mbps são 96 vezes 64 kbps mais 168 kbps de carga adicional. Esse serviço pode ser usado como um único serviço para transmissões a 6.312 Mbps; por outro lado, ele pode ser usado para multiplexar 4 canais DS-1, 96 canais DS-0, ou alguma combinação desses tipos de serviços.
- O **DS-3** é um serviço de 44.376-Mbps; 44.376 Mbps são 672 vezes 64 kbps mais 1.368 Mbps de carga adicional. Esse serviço pode ser usado como um único serviço para transmissões a 44.376 Mbps; por outro lado, ele pode ser usado para multiplexar 7 canais DS-2, 28 canais DS-1, 672 canais DS-0, ou alguma combinação desses tipos de serviços. A aplicação comercial desse serviço é conhecida como uma linha **T-3**.
- O **DS-4** é um serviço de 274.176-Mbps; 274.176 Mbps são 4.032 vezes 64 kbps mais 16.128 Mbps de carga adicional. Ele pode ser usado para multiplexar 6 canais DS-3, 42 canais DS-2, 168 canais DS-1, 4.032 canais DS-0 ou uma combinação desses tipos de serviços.

Multiplexação por divisão de tempo estatística

Conforme vimos na seção anterior, cada entrada na TDM síncrona tem uma parcela de tempo reservada no quadro de saída. Isso pode ser ineficiente se algumas linhas de entrada não tiverem dados para enviar. Na multiplexação por divisão de tempo estatística, as parcelas são alocadas dinamicamente para melhorar a eficiência no uso da largura de banda. Apenas quando uma linha de entrada tiver uma quantidade de dados equivalente a uma parcela de tempo, ela recebe uma parcela

de tempo no quadro de saída. Na multiplexação estatística, o número de parcelas de tempo em cada quadro é geralmente menor do que o número de linhas de entrada. O multiplexador verifica cada linha de entrada no estilo *round-robin*, ou seja, ele aloca uma parcela de tempo para uma linha de entrada caso a linha tenha dados para enviar e, caso contrário, ignora essa linha e verifica a próxima. A Figura 7.52 mostra um exemplo de TDM síncrona e de TDM estatística. No primeiro caso, algumas parcelas de tempo ficam vazias porque a linha correspondente não tem dados para enviar. Nesse caso, por outro lado, nenhuma parcela de tempo fica vazia enquanto houver dados para serem enviados por qualquer linha de entrada.

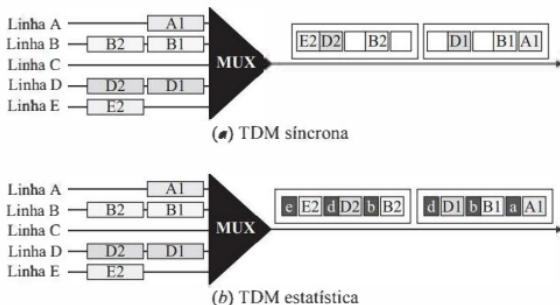



Figura 7.52 Comparação das parcelas de tempo nos métodos de TDM.

A Figura 7.52 também mostra uma grande diferença entre as parcelas de tempo na TDM síncrona e na TDM estatística. Na saída da TDM síncrona, cada parcela de tempo é ocupada apenas pelos dados; já na TDM estatística, uma parcela de tempo precisa transportar os dados juntamente com o seu endereço de destino. Na TDM síncrona, não é necessário usar endereçamento; a sincronização e as relações pré-atribuídas entre as entradas e saídas servem como endereço. Sabemos, por exemplo, que a entrada 1 vai sempre para a saída 2. Se o multiplexador e o demultiplexador estiverem sincronizados, isso é garantido. Na multiplexação estatística, não há uma relação fixa entre as entradas e as saídas, pois as parcelas de tempo não são pré-atribuídas ou reservadas. Precisamos incluir o endereço do receptor junto com cada parcela de tempo para mostrar onde ela deve ser entregue. Em sua forma mais simples, o endereçamento pode consistir em *n bits* para definir *N* linhas de saída diferentes, com $n = \log_2 N$. Por exemplo, para oito linhas de saída diferentes, precisamos de um endereço de 3 *bits*.

7.4.2 Espalhamento Espectral

A multiplexação combina sinais de diversas fontes para conseguir uma maior eficiência no uso da largura de banda; em um enlace, ela é dividida entre as fontes. No **Espalhamento Espectral** (SS – Spread Spectrum), também combinamos sinais de diferentes fontes para preencher uma largura de banda maior, mas nossos objetivos são um pouco diferentes. O espalhamento espectral foi projetado para ser usado em aplicações sem fios (LANs e WANs sem fios). Nesse tipo de aplicações, temos algumas preocupações mais importantes que a eficiência no uso da largura de banda. Em aplicações sem fios, todas as estações usam o ar (ou o vácuo) como meio de comunicação. As estações devem ser capazes de compartilhar esse meio sem que haja interceptação dos dados por um intruso e sem que a comunicação fique sujeita a interferências geradas por uma entidade mal-intencionada (em operações militares, por exemplo).

Para atingir esses objetivos, as técnicas de espalhamento espectral adicionam redundância aos dados; elas espalham o espectro original exigido por cada estação. Se a largura de banda exigida por cada estação for B, o espalhamento espectral eleva a demanda para B_{ss}, de forma que $B_{ss} >> B$. A largura de banda expandida permite que a fonte coloque sua mensagem em um envelope de proteção para criar uma transmissão mais segura. Uma analogia é o envio de um presente delicado e caro. Podemos colocá-lo em uma caixa especial para impedir que ele seja danificado durante o transporte, e podemos usar um serviço de entrega de melhor qualidade para garantir a proteção do pacote.

A Figura 7.53 mostra a ideia do espalhamento espectral, que atinge seus objetivos por meio de dois princípios:

1. A largura de banda alocada para cada estação precisa ser muito maior do que a necessária. Isto permite a introdução de redundância.
2. O aumento da largura de banda do valor original B para o valor B_{ss} deve ser feito por meio de um processo que independe do sinal original. Em outras palavras, o processo de espalhamento ocorre após o sinal ser criado pela fonte.

Figura 7.53 Espalhamento espectral.

Após o sinal ter sido criado pela fonte, o processo de espalhamento usa um código de espalhamento e expande a largura de banda. A figura mostra a largura de banda original B e a largura de banda expandida B_{ss}. O código de espalhamento consiste em uma série de números de aparência aleatória, mas que apresentam, na verdade, um padrão.

Existem duas técnicas para espalhar a largura de banda: Espalhamento Espectral por Saltos em Frequências (FHSS – Frequency Hopping Spread Spectrum) e Espalhamento Espectral por Sequência Direta (DSSS – Direct Sequence Spread Spectrum).

Espalhamento Espectral por Saltos em Frequências

A técnica de **Espalhamento Espectral por Saltos em Frequências** (FHSS – Frequency Hopping Spread Spectrum) utiliza M frequências portadoras diferentes que são moduladas pelo sinal de origem. Em certo instante, o sinal modula uma frequência portadora; no instante seguinte, o sinal modula outra frequência portadora. Embora a modulação seja feita usando uma frequência portadora de cada vez, as M frequências são utilizadas quando se observa um longo período de tempo. A largura de banda ocupada por uma fonte após o espalhamento é $B_{FHSS} >> B$. A Figura 7.54 mostra a estrutura geral do FHSS.

Um **gerador pseudoaleatório de códigos**, denominado **ruído pseudoaleatório** (RP), cria uma sequência de k *bits* para cada **período de salto** T_h. A tabela de frequências usa essa sequência para determinar a frequência a ser utilizada no período de salto atual e passa o resultado para o sintetizador de frequências. Este cria um sinal de portadora com aquela frequência, enquanto o sinal original modula o sinal de portadora. Considere que decidimos usar oito frequências de salto, um número é extremamente baixo para aplicações reais, sendo usado apenas para fins de ilustração. Nesse caso, M vale 8 e k vale 3. O gerador de código pseudoaleatório criará oito sequências de *bits* diferentes. Essas sequências são mapeadas para oito frequências distintas na tabela de frequências.

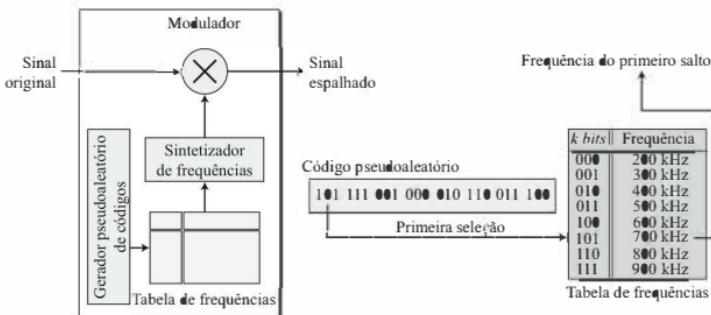



Figura 7.54 Espalhamento Espectral por Saltos em Frequências (FHSS).

A sequência para essa estação é 101, 111, 001, 000, 010, 011, 100. Perceba que a sequência é pseudoaleatória; ela é repetida após oito saltos. Isto significa que no período de salto 1, o padrão é 101. A frequência selecionada vale 700 kHz; o sinal da fonte modula essa frequência portadora. O segundo padrão de *k bits* escolhido é 111, que seleciona a portadora de 900-kHz; a oitava sequência é 100, e, portanto, a frequência é 600 kHz. Depois de oito saltos, o padrão se repete, começando novamente em 101. A Figura 7.55 mostra como o sinal salta de portadora em portadora. Consideramos que a largura de banda necessária para o sinal original é de 100 kHz.

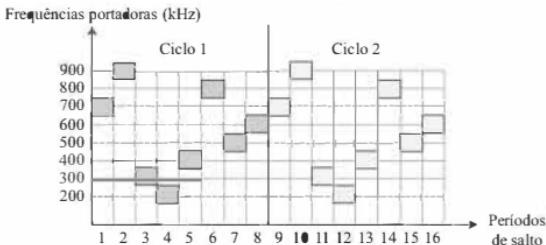



Figura 7.55 Ciclos do FHSS.

Pode-se demonstrar que esse esquema é capaz de satisfazer as metas mencionadas anteriormente. Se existirem muitas sequências de *k bits* e se o período de salto for curto, um emissor e um receptor podem se comunicar com privacidade. Se um intruso tentar interceptar o sinal transmitido, ela só será capaz de acessar uma pequena parte dos dados, pois não conhece a sequência de espalhamento de forma a se adaptar rapidamente para o próximo salto. O esquema também apresenta um efeito anti-interferência. Um emissor malicioso pode ser capaz de transmitir ruído para causar interferência no sinal durante um período de salto (aleatoriamente), mas isso não pode ser feito para todo o período.

Compartilhamento de largura de banda

Se o número de frequências de salto for M, podemos multiplexar M canais em um usando a mesma largura de banda B_{ss}. Isto é possível porque cada estação utiliza apenas uma frequência em cada

período de salto; as outras $M - 1$ frequências podem ser usadas por outras $M - 1$ estações. Em outras palavras, M estações diferentes podem usar a mesma B_{ss} se uma técnica de modulação apropriada, como o FSK multinível (MFSK), for usada. O FHSS é semelhante à técnica de FDM, conforme mostra a Figura 7.56.

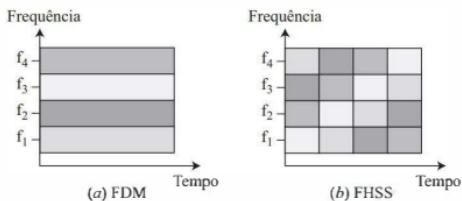



Figura 7.56 Compartilhamento de largura de banda.

A Figura 7.56 mostra um exemplo de quatro canais usando FDM e quatro canais usando FHSS. Na FDM, cada estação utiliza $1/M$ da largura de banda, mas a alocação é fixa; no FHSS, cada estação também utiliza $1/M$ da largura de banda, porém a alocação é alterada salto a salto.

Espalhamento Espectral por Sequência Direta

A técnica de **Espalhamento Espectral por Sequência Direta** (DSSS – Direct Sequence Spread Spectrum) também expande a largura de banda do sinal original, mas o processo é diferente. No DSSS, substituímos cada *bit* de dados por *n bits* usando um código de espalhamento. Em outras palavras, a cada *bit* é atribuído um código de *n bits*, denominado uma ficha, onde a taxa de fichas corresponde a *n* vezes a taxa de *bits* de dados. A Figura 7.57 ilustra o conceito do DSSS.

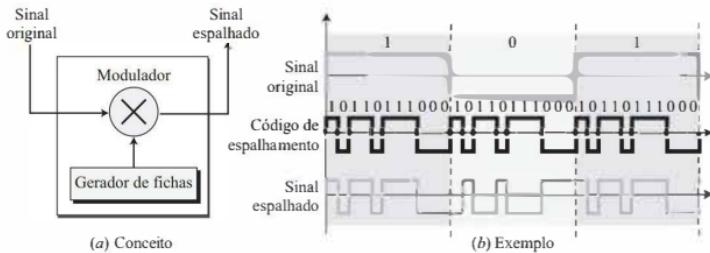



Figura 7.57 DSSS

A título de exemplo, consideremos a sequência usada em uma LAN sem fio, a famosa **sequência de Barker**, na qual *n* vale 11, e o sinal original e as fichas no gerador de fichas usem a codificação NRZ polar. A Figura 7.57 mostra as fichas e o resultado da multiplicação dos dados originais pelas fichas para obter o sinal espalhado.

Na Figura 7.57, o código de espalhamento consiste em 11 fichas com a sequência 10110111000 (neste caso). Se a taxa de transferência do sinal original for *N*, a taxa de transferência do sinal

espalhado vale 11*N*. Isto significa que a largura de banda necessária para o sinal espalhado é 11 vezes maior do que a largura de banda do sinal original. O sinal espalhado é capaz de proporcionar privacidade à comunicação se o invasor não souber o código. Ele também pode fornecer imunidade contra interferências se cada estação utilizar um código diferente.

7.5 MEIOS DE TRANSMISSÃO

Discutimos diversas questões relacionadas à camada física neste capítulo. Nesta seção, abordamos os meios de transmissão, que ficam, na realidade, localizados abaixo da camada física e são controlados diretamente por ela. Podemos dizer que os meios de transmissão pertencem à camada zero. A Figura 7.58 mostra a localização dos meios de transmissão em relação à camada física.

Figura 7.58 Meios de transmissão e a camada física.

Um **meio de transmissão** pode ser definido de forma ampla como qualquer coisa capaz de transportar informações de uma origem até um destino. Por exemplo, o meio de transmissão no caso de duas pessoas tendo uma conversa durante um jantar é o ar, que também pode ser usado para transmitir mensagens na forma de um sinal de fumaça ou do sinal luminoso de um semáforo. Para uma mensagem escrita, o meio de transmissão pode ser um carteiro, um caminhão ou um avião.

No contexto de comunicação de dados, as definições de informação e de meio de transmissão são mais específicas. O meio de transmissão é normalmente um espaço aberto, cabo metálico ou cabo de fibra óptica. A informação costuma ser um sinal resultante da conversão de dados representados em outra forma.

A energia eletromagnética, uma combinação de campos elétricos e magnéticos oscilando ortogonalmente em relação um ao outro, inclui ondas de rádio, luz infravermelha, luz visível, luz ultravioleta, raios X, raios gama e raios cósmicos. Cada uma dessas ondas compõe uma porção do **espectro eletromagnético**. Nem todas as porções do espectro são utilizáveis atualmente nas telecomunicações, entretanto. Além disso, existe um número limitado de tipos de meios de comunicação utilizados para aproveitar as porções úteis do espectro.

Nas telecomunicações, os meios de transmissão podem ser divididos em duas grandes categorias: guiados e não guiados. Meios guiados incluem cabos de par trançado, coaxiais e de fibra óptica. O meio não guiado é o espaço aberto.

7.5.1 Meios guiados

Os **meios guiados**, que são aqueles que fornecem um canal para a condução do sinal de um dispositivo a outro, incluem **cabo de par trançado, cabo coaxial** e **cabo de fibra óptica**. Um sinal viajando por qualquer desses meios é direcionado e confinado pelos limites físicos do meio. Cabos de par trançado e coaxiais usam condutores metálicos (feitos de cobre) para receber e transportar sinais na forma de corrente elétrica. Cabos de fibra óptica são cabos que recebem e transportam sinais na forma de luz.

Cabo de par trançado

Um cabo de par trançado consiste em dois condutores (normalmente feitos de cobre), cada um com seu próprio isolamento de plástico, trançados um ao redor do outro, conforme mostrado na Figura 7.59.

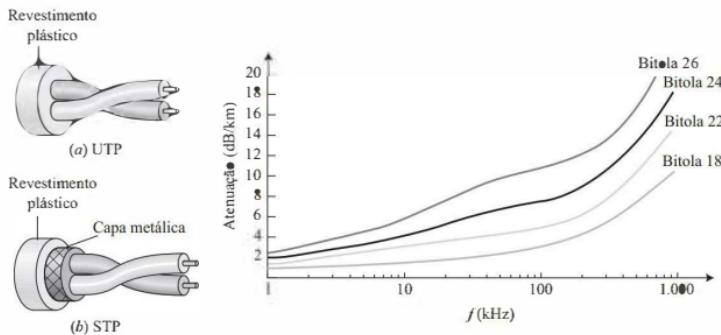



Figura 7.59 Cabo de par trançado.

Um dos fios é usado para transportar sinais provenientes do receptor, enquanto o outro é usado apenas como um valor de referência (terra). O receptor usa a diferença entre os dois.

Além do sinal proveniente do remetente, a interferência (ruído) e a diafonia (interferência cruzada) podem afetar ambos os fios e criar sinais indesejados. Se os dois fios forem paralelos, o efeito desses sinais indesejáveis não será o mesmo em ambos os fios, pois eles ficam em locais diferentes em relação às fontes de ruído ou de diafonia. Isso resulta em uma diferença no sinal que chega ao receptor. Ao trançar os pares, um equilíbrio é mantido. O cabo de par trançado de modo geral usado em comunicações é conhecido como cabo **Par Trançado sem Blindagem** (UTP – Unshielded Twisted-Pair). A IBM produziu também uma versão do cabo de par trançado para seu próprio uso denominado cabo **Par Trançado Blindado** (STP – Shielded Twisted-Pair). O cabo STP apresenta uma cobertura na forma de uma folha ou malha trançada de metal, a qual envolve cada par de condutores isolados. Embora o invólucro de metal melhore a qualidade do cabo, impedindo a penetração de sinais de ruído ou diafonia, ele torna o cabo mais volumoso e mais caro.

Desempenho

Uma maneira de medir o desempenho do cabo de par trançado consiste em comparar a atenuação em função da frequência e da distância. Um cabo de par trançado permite a passagem de uma ampla gama de frequências. No entanto, a Figura 7.59 também mostra que, com o aumento da frequência, a atenuação, medida em decibéis por quilômetro (dB/km), aumenta consideravelmente para frequências acima de 100 kHz. Perceba que a **bitola**, também conhecida como calibre, é uma medida da espessura do fio (inversamente proporcional a ela).

Aplicações

Cabos de par trançado são usados em linhas telefônicas para fornecer canais de voz e de dados. O lacete local – a linha que conecta os assinantes à central telefônica – normalmente é composto por cabos de par trançado sem blindagem. As linhas DSL que são usadas pelas companhias telefônicas para fornecer conexões de dados de alta velocidade também utilizam a elevada largura de banda dos cabos de par trançado sem blindagem. Redes locais, como 10Base-T e 100Base-T, também utilizam cabos de par trançado. Discutimos essas redes no Capítulo 5.

Cabo coaxial

Os cabos coaxiais (ou *coax*) transportam sinais de faixas de frequências mais altas do que aquelas transportadas em cabos de par trançado, em parte porque os dois meios são construídos de forma bastante diferente. Em vez de apresentarem dois fios, os cabos coaxiais têm um núcleo central condutor feito de fios (normalmente cobre) trançados ou maciços recobertos por uma capa isolante. Essa capa é, por sua vez, revestida por um condutor externo metálico na forma de uma folha, malha ou uma combinação de ambas. O invólucro externo metálico serve como uma proteção contra ruído e como um segundo condutor, completando o circuito. Esse condutor externo também é revestido por uma capa isolante e o cabo todo é protegido por uma cobertura plástica (ver Figura 7.60).

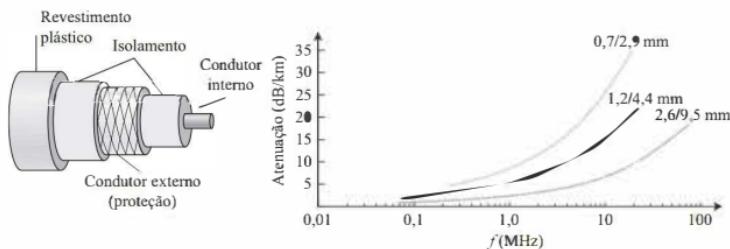



Figura 7.60 Cabo coaxial.

Desempenho

Assim como fizemos com cabos de par trançado, podemos medir o desempenho de um cabo coaxial. Podemos notar na Figura 7.60 que a atenuação é muito maior no cabo coaxial do que no cabo de par trançado. Em outras palavras, embora o cabo coaxial apresente uma largura de banda muito mais elevada, o sinal se enfraquece rapidamente e requer o uso frequente de repetidores.

Aplicações

Os cabos coaxiais foram amplamente utilizados em redes telefônicas analógicas nas quais uma única rede coaxial podia transportar 10.000 sinais de voz. Mais tarde, eles foram utilizados em redes telefônicas digitais, nas quais um único cabo coaxial era capaz de transportar dados digitais a uma taxa de até 600 Mbps. Entretanto, atualmente, os cabos coaxiais usados nas redes telefônicas foram, em sua maioria, substituídos por cabos de fibra óptica.

Redes de TV a cabo também usam cabos coaxiais; nas tradicionais, toda a rede era composta por cabos coaxiais. Mais tarde, porém, os provedores de TV a cabo substituíram a maioria de seus cabos por cabos de fibra óptica; redes híbridas usam cabos coaxiais apenas nas bordas da rede, próximo às instalações do consumidor. As aplicações de TV a cabo usam cabos coaxiais RG-59.

Outra aplicação comum de cabos coaxiais é nas LANs Ethernet tradicionais (ver Capítulo 5). Devido à sua elevada largura de banda e, consequentemente, à alta taxa de transferência de dados, os cabos coaxiais foram escolhidos para a transmissão digital nas primeiras LANs Ethernet. As redes do tipo Thick Ethernet (10BASE5), também conhecidas como redes de cabo Ethernet grosso, usam conectores especiais.

Cabo de fibra óptica

Um cabo de fibra óptica é feito de vidro ou plástico e transmite sinais na forma de luz. Para entender as fibras ópticas, primeiro precisamos explorar vários aspectos da natureza da luz.

A luz viaja em linha reta enquanto estiver atravessando uma única substância uniforme. Se um raio de luz estiver viajando por uma substância e de repente entrar em outra (de densidade diferente), o raio muda de direção. A Figura 7.61 mostra como um raio de luz muda de direção quando ele passa de uma substância mais densa para outra menos densa.

Figura 7.61 Alteração na direção de um raio de luz.

Conforme mostrado na figura, se o **ângulo de incidência** *I* (o ângulo que o raio faz com a linha perpendicular à interface entre as duas substâncias) for menor do que o **ângulo crítico**, o raio refrata e se aproxima da superfície. Se o ângulo de incidência for igual ao ângulo crítico, a luz refratada passa a viajar paralelamente à interface entre os dois meios. Se o ângulo for maior do que o ângulo crítico, o raio é refletido (ele retoma) e continua a viajar na substância mais densa. Perceba que o ângulo crítico é uma propriedade da substância, e o seu valor difere de uma substância para outra.

As fibras ópticas usam a reflexão para guiar a luz através de um canal. Um **núcleo** de vidro ou de plástico é cercado por um **revestimento** de vidro ou plástico menos denso. A diferença de densidade dos dois materiais deve ser tal que um feixe de luz atravessando o núcleo seja refletido pelo revestimento em vez de ser refratado através dele. Ver Figura 7.62.

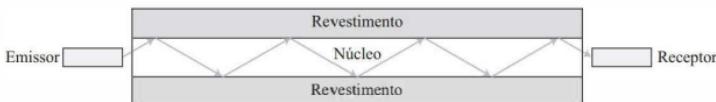



Figura 7.62 Fibra óptica.

Modos de propagação

A tecnologia atual fornece suporte a dois modos (multimodo e modo único) de propagação da luz ao longo dos canais ópticos, cada um deles exigindo uma fibra com características físicas diferentes. A propagação multimodo pode ser implementada de duas formas: índice degrau (*step-index*) ou índice gradual (*grade-index*). A propagação multimodo é assim denominada porque, nesse caso, múltiplos feixes provenientes de uma fonte de luz atravessam o núcleo por caminhos distintos. A forma como eles se movem dentro do cabo depende da estrutura do núcleo, conforme mostra a Figura 7.63.

Nas **fibras multimodo de índice degrau**, a densidade do núcleo permanece constante do centro até as bordas. Um feixe de luz move-se através dessa densidade constante em linha reta até atingir a interface entre o núcleo e o revestimento. Nela, ocorre uma mudança abrupta devido à menor densidade do revestimento; isto altera o ângulo da direção do feixe. O termo *índice degrau* refere-se à rapidez dessa mudança, a qual contribui para a distorção do sinal à medida que ele viaja pela fibra.

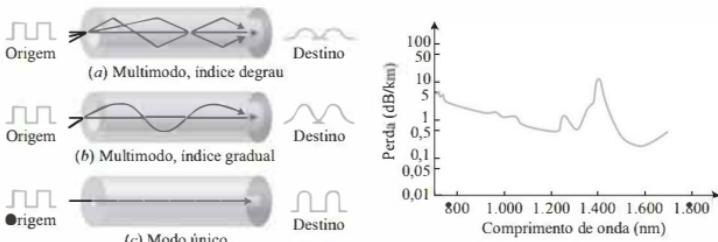



Figura 7.63 Modos

Um segundo tipo de fibra, denominado **fibra multimodo de índice gradual**, diminui essa distorção do sinal ao longo do cabo. A palavra *índice* aqui se refere ao índice de refração. Conforme vimos anteriormente, o índice de refração tem relação com a densidade do meio. Uma fibra de índice gradual, portanto, é uma fibra cuja densidade varia. A densidade é mais elevada no centro do núcleo e diminui gradualmente conforme nos aproximamos das bordas da fibra.

A propagação em modo único utiliza fibras de índice degrau e uma fonte de luz altamente focada que limita os ângulos dos feixes a um pequeno intervalo, de modo que todos os ângulos ficam próximos da horizontal. As **fibras de modo único** em si são fabricadas com um diâmetro muito menor do que aquele das fibras multimodo, e com uma densidade (índice de refração) substancialmente menor. A diminuição na densidade resulta em um ângulo crítico que fica suficientemente próximo de 90° para fazer com que a propagação dos feixes seja quase horizontal. Neste caso, a propagação dos diferentes feixes permanece quase idêntica e os atrasos são insignificantes. Todos os feixes chegam ao destino "juntos" e podem ser recombinados com reduzida distorção do sinal.

Desempenho

O gráfico da atenuação *versus* o comprimento de onda apresentado na Figura 7.63 também mostra um fenômeno muito interessante que ocorre em cabos de fibra óptica. A curva de atenuação, apresentada no gráfico, é mais achatada do que no caso de cabos de par trançado ou cabos coaxiais. O desempenho é tal que precisamos de menos (na verdade, 10 vezes menos) repetidores quando usamos cabos de fibra óptica.

Aplicações

Os cabos de fibra ótica são frequentemente encontrados em redes de *backbone* porque sua ampla largura de banda apresenta uma melhor relação custo-benefício. Atualmente, com a multiplexação por divisão de comprimento de onda (WDM), podemos transmitir dados a uma velocidade de 1.600 Gbps. A rede SONET, que discutimos no Capítulo 5, fornece tal *backbone*.

Algumas empresas de TV a cabo usam uma combinação de fibras ópticas e cabos coaxiais, criando, assim, uma rede híbrida. As fibras ópticas constituem a estrutura do *backbone*, enquanto os cabos coaxiais fornecem as conexões até as instalações do usuário. Essa é uma configuração econômica dado que a largura de banda necessária pelos usuários finais é estreita demais para justificar o uso de fibras ópticas.

Redes locais, como 100Base-FX (Fast Ethernet) e 1000Base-X também usam cabos de fibra óptica. A operação da Ethernet 10-Gigabit também utiliza cabos de fibra óptica.

7.5.2 Meios não guiados: Sem fios

Os **meios não guiados** transportam ondas eletromagnéticas sem usar um condutor físico. Esse tipo de comunicação é muitas vezes conhecido como a *comunicação sem fios* ou *comunicação wireless*. Os sinais são normalmente transmitidos na forma de *broadcast* através do espaço aberto e, portanto, ficam disponíveis para quem quer que tenha um dispositivo capaz de captá-los.

A Figura 7.64 mostra a parte do espectro eletromagnético, na faixa de 3 kHz a 900 THz, usada para a comunicação sem fios.

Figura 7.64 Espectro eletromagnético usado na comunicação sem fios.

Sinais não guiados podem viajar da origem até o destino de diversas formas: propagação terrestre, propagação celeste e propagação em visada direta.

Na **propagação terrestre**, as ondas de rádio viajam na porção mais baixa da atmosfera, junto à Terra. Esses sinais de baixa frequência se propagam em todas as direções a partir da antena de transmissão e seguem a curvatura do planeta. A distância alcançada depende da quantidade de energia no sinal: quanto maior a potência, maior é a distância. Na **propagação celeste** (também conhecida como **propagação ionosférica**), ondas de rádio de frequência mais alta são irradiadas para o alto, chegando até a ionosfera (camada da atmosfera onde as partículas existem como íons), onde são refletidas de volta para a Terra*. Esse tipo de transmissão permite alcançar maiores distâncias com uma menor potência de saída. Na **propagação em visada direta**, sinais de frequência muito alta são transmitidos em linha reta, diretamente de antena a antena. As antenas devem ser direcionais, voltadas uma para a outra, e altas o suficiente ou estarem próximas o suficiente de modo a não serem afetadas pela curvatura da terra. O uso de propagação em visada direta é complicado, porque não é possível obter transmissões de rádio completamente focadas.

A porção do espectro eletromagnético definida como ondas e micro-ondas de rádio é dividida em oito faixas, denominadas *bandas*, cada uma regulamentada por autoridades governamentais. Essas bandas são classificadas de Frequência Muito Baixa (VLF – Very Low Frequency) a Frequência Extremamente Alta (EHF – Extremely High Frequency). A Tabela 7.1 lista as bandas, suas faixas de valores, os métodos de propagação correspondentes e algumas de suas aplicações.

Podemos dividir a transmissão sem fios em três grandes grupos: ondas de rádio, micro-ondas e ondas infravermelhas.

Ondas de rádio

Embora não exista uma demarcação exata que diferencie ondas de rádio e micro-ondas, as ondas eletromagnéticas com frequências variando entre 3 kHz e 1 GHz são normalmente denominadas **ondas de rádio;** ondas com frequências variando entre 1 e 300 GHz são denominadas **micro-ondas**. Entretanto, o comportamento das ondas, e não suas frequências, é um melhor critério para sua classificação. As ondas de rádio, em sua maior parte, são omnidirecionais. Quando uma antena transmite ondas de rádio, elas se propagam em todas as direções. Isto significa que as antenas

* N. de T.: Somente ondas em determinadas faixas de frequência são refletidas por diferentes camadas da ionosfera. Frequências em faixas muito altas podem atravessar a ionosfera e continuar se propagando no espaço (vácuo).

Tabela 7.1 Bandas

Banda	Faixa	Propagação	Aplicação
VLF (*Very Low Frequency*, ou Frequência Muito Baixa)	3-30 kHz	Terrestre	Rádio de longa distância
LF (*Low Frequency*, ou Frequência Baixa)	30-300 kHz	Terrestre	Radiobaliza*
MF (*Middle Frequency*, ou Frequência Média)	300 kHz-3 MHz	Celeste	Rádio AM
HF (*High Frequency*, ou Frequência Alta)	3-30 MHz	Celeste	Radioamador, comunicação de navios/aeronaves
VHF (*Very High Frequency*, ou Frequência Muito Alta)	30-300 MHz	Celeste e visada direta	TV VHF, rádio FM
UHF (*Ultrahigh Frequency*, ou Frequência Ultra Alta)	300 MHz-3 GHz	Visada direta	TV UHF, telefonia celular, *pagers*, satélite
SHF (*Superhigh Frequency*, ou Frequência Super Alta	3-30 GHz	Visada direta	Comunicação via satélite
EHF (*Extremely High Frequency*, ou Frequência Extremamente Alta)	30-300 GHz	Visada direta	Radar, satélite

emissora e receptora não precisam estar alinhadas. Uma antena emissora emite ondas que podem ser recebidas por qualquer antena receptora. A propriedade omnidirecional também tem uma desvantagem. As ondas de rádio transmitidas por uma antena são suscetíveis a interferências causadas por outra antena que esteja enviando sinais na mesma frequência ou na mesma banda.

As ondas de rádio, em particular as ondas que apresentam propagação celeste, podem viajar longas distâncias tornando-se boas candidatas para a transmissão de mensagens *broadcast* em longa distância, como no caso do rádio AM.

As ondas de rádio, em particular aquelas de frequências baixas e médias, podem atravessar paredes. Essa característica pode ser tanto uma vantagem como uma desvantagem. Ela é uma vantagem porque, por exemplo, um rádio AM dentro de um edifício é capaz de receber os sinais. É uma desvantagem porque não se pode isolar uma comunicação de modo que ela fique confinada apenas ao interior ou ao exterior de um edifício. A banda das ondas de rádio é relativamente estreita, ficando ligeiramente inferior a 1 GHz, quando comparada à banda de micro-ondas. Quando essa banda é dividida em sub-bandas, estas também são estreitas, levando a uma reduzida taxa de transmissão de dados para comunicações digitais.

Aproximadamente toda a banda é regulada pelas autoridades** (por exemplo, a FCC nos Estados Unidos). O uso de qualquer parte da banda requer a permissão das autoridades.

Micro-ondas

As ondas eletromagnéticas com frequências entre 1 e 300 GHz são denominadas micro-ondas. Micro-ondas são unidirecionais. Quando uma antena transmite micro-ondas, elas podem ser concentradas em uma região estreita. Isto significa que as antenas emissora e receptora precisam estar

* N. de T.: Transmissão que serve de auxílio de navegação para determinar a direção ou a posição.

** N. de T.: No Brasil, o órgão responsável por regular e fiscalizar o setor de telecomunicações é a Agência Nacional de Telecomunicações (Anatel).

alinhadas. A propriedade unidirecional tem uma vantagem óbvia. Um par de antenas pode ser alinhado sem interferir com outro par de antenas alinhadas. A seguir, são descritas algumas características da propagação de micro-ondas:

- A propagação das micro-ondas é do tipo visada direta. Como as torres com as antenas montadas precisam ser diretamente visíveis umas pelas outras, torres que estejam muito afastadas precisam ser muito altas. A curvatura da Terra, bem como outros obstáculos, não permite que duas torres baixas se comuniquem usando micro-ondas. Repetidores são muitas vezes necessários para permitir a comunicação de longa distância.
- Micro-ondas de frequência muito alta não são capazes de atravessar paredes. Essa característica pode ser uma desvantagem se os receptores estiverem dentro de edifícios.
- A banda de micro-ondas é relativamente larga, chegando a quase 299 GHz. Portanto, podem ser atribuídas sub-bandas mais largas a diferentes entidades, permitindo uma elevada taxa de transferência de dados.
- O uso de algumas partes da banda requer permissão das autoridades.

Aplicações

Micro-ondas, devido às suas propriedades unidirecionais, são muito úteis quando a comunicação *unicast* (um para um) se faz necessária entre o emissor e o receptor. Elas são usadas em telefones celulares, redes de satélite e redes locais sem fios.

Infravermelho

As **ondas de infravermelho**, com frequências de 300 GHz a 400 THz (comprimentos de onda de 1 mm a 770 nm), podem ser usadas para comunicações de curto alcance. As ondas de infravermelho, por apresentarem frequências elevadas, não são capazes de atravessar paredes. Essa característica vantajosa previne a interferência entre um sistema e outro; um sistema de comunicação de curto alcance em uma sala não pode ser afetado por outro sistema na sala ao lado. Quando usamos um controle remoto por infravermelho, não interferimos na utilização do controle remoto de nossos vizinhos. No entanto, essa mesma característica torna inúteis os sinais infravermelhos para a comunicação de longo alcance. Além disso, não podemos usar ondas de infravermelho no exterior de edifícios porque os raios solares contêm ondas de infravermelho que podem interferir na comunicação.

7.6 MATERIAL DO FINAL DO CAPÍTULO

7.6.1 Leitura recomendada

Para mais detalhes sobre os assuntos discutidos neste capítulo, recomendamos os seguintes livros. Os itens entre colchetes [...] referem-se à lista de referências no fim do texto.

Livros

Diversos livros cobrem os temas discutidos nestes capítulo, incluindo [Pea 92], [Cou 01], [Ber 96], [Hsu 03], [Spi 74], [Sta 04], [Tan 03], [G & W 04], [SSS 05], [BEL 01] e [Max 99].

7.6.2 Termos-chave

- 2-Binário, 1-Quaternário (2B1Q)
- 4-Binário/5-Binário (4B/5B)
- 8-Binário, 6-Ternário (8B6T)
- amostra e retém (*sample and hold*)
- amostragem
- amplitude de pico

- ângulo crítico
- ângulo de incidência
- atenuação
- banda de guarda
- Bipolar com Substituição de 8 Zeros (B8ZS – Bipolar with 8-Zero Substitution)
- Bipolar de Alta Densidade de 3 Zeros (HDB3 – High-Density Bipolar 3-Zero)
- cabo coaxial
- cabo de fibra óptica
- cabo de par trançado
- canal passa-baixas
- canal passa-faixa
- capacidade de Shannon
- codificação bipolar
- codificação de linha
- codificação de bloco
- conversão analógico-analógico
- conversão analógico-digital
- conversão digital-analógico
- conversão digital-digital
- dados analógicos
- dados digitais
- decibel (dB)
- diagrama de constelação
- digitalização
- distorção
- embaralhamento (*scrambling*)
- Espalhamento Espectral (SS – Spread Spectrum)
- Espalhamento Espectral por Saltos em Frequências (FHSS – Frequency-Hopping Spread Spectrum)
- Espalhamento Espectral por Sequência Direta (DSSS – Direct Sequence Spread Spectrum)
- espectro eletromagnético
- fase
- fibra de modo único
- fibra multimodo de índice degrau
- fibra multimodo de índice gradual
- frequência
- interferência cruzada (diafonia)
- Inversão Alternada de Marcas (AMI – Alternate Mark Inversion)
- *jitter*
- largura de banda
- linha T
- Manchester
- Manchester diferencial
- meio de transmissão
- meio guiado
- meio não guiado
- Modulação de Amplitude (AM – Amplitude Modulation)
- Modulação de Amplitude de Pulso (PAM – Pulse Amplitude Modulation)
- Modulação de Fase (PM – Phase Modulation)
- Modulação de Frequência (FM – Frequency Modulation)
- modulação delta
- Modulação por Amplitude em Quadratura (QAM – Quadrature Amplitude Modulation)
- Modulação por Chaveamento de Amplitude (ASK – Amplitude Shift Keying)
- Modulação por Chaveamento de Fase (PSK – Phase Shift Keying)
- Modulação por Chaveamento de Frequência (FSK – Frequency Shift Keying)
- Modulação por Código de Pulsos (PCM – Pulse Code Modulation)
- multiplexação
- Multiplexação por Divisão de Comprimento de Onda (WDM – Wavelength-Division Multiplexing)
- Multiplexação por Divisão de Frequência (FDM – Frequency-Division Multiplexing)
- Multiplexação por Divisão de Tempo (TDM – Time-Division Multiplexing)
- Par Trançado Blindado (STP – Shielded Twisted-Pair)
- Par Trançado sem Blindagem (UTP – Unshielded Twisted-Pair)
- período
- propagação celeste
- propagação em visada direta
- propagação terrestre
- pseudoternário
- Relação Sinal/Ruído (S/R)
- representação no domínio da frequência
- representação no domínio do tempo
- Retorno ao Zero (RZ – Return-to-Zero)
- ruído
- ruído pseudoaleatório (RP)
- Sequência de Barker
- serviço de sinal digital (DS – digital signal)
- sinal analógico
- sinal aperiódico
- sinal composto
- sinal digital
- sinal periódico
- taxa de amostragem
- taxa de *bits*
- taxa de *bits* de Nyquist
- TDM estatística

- TDM síncrona
- teorema de Nyquist
- transmissão em banda base
- vazão

7.6.3 Resumo

Os dados devem ser transformados em sinais eletromagnéticos para serem transmitidos. Os dados analógicos são contínuos e assumem valores contínuos. Dados digitais apresentam estados discretos e assumem valores discretos. Os sinais analógicos podem apresentar um número infinito de valores em um intervalo; os digitais podem assumir apenas um número limitado de valores. No contexto de comunicação de dados, geralmente usamos sinais analógicos periódicos e sinais digitais aperiódicos.

A conversão digital-digital envolve três técnicas: codificação de linha, codificação de bloco e embaralhamento. A codificação de linha é um processo de conversão de dados digitais em um sinal digital. A codificação de bloco fornece redundância para fornecer mecanismos de sincronização e detecção de erros inerentes à transmissão. A técnica mais comum para transformar um sinal analógico em dados digitais (digitalização) é conhecida como Modulação por Código de Pulsos (PCM – Pulse Code Modulation).

A conversão digital-analógico consiste no processo de transformar uma das características de um sinal analógico com base na informação contida nos dados digitais. A conversão digital-analógico pode ser realizada de várias maneiras: Modulação por Chaveamento de Amplitude (ASK – Amplitude Shift Keying), Modulação por Chaveamento de Frequência (FSK – Frequency Shift Keying) e Modulação por Chaveamento de Fase (Modulação PSK – Phase Shift Keying). A Modulação por Amplitude em Quadratura (modulação QAM – Quadrature Amplitude Modulation) combina ASK e PSK. A conversão analógico-analógico pode ser realizada de três formas: Modulação de Amplitude (AM – Amplitude Modulation), Modulação de Frequência (FM – Frequency Modulation) e Modulação de Fase (PM – Phase Modulation).

A utilização de banda consiste no uso da largura de banda disponível para atingir objetivos específicos. Pode-se obter maior eficiência nesse uso aplicando-se técnicas de multiplexação; privacidade e resistência a interferências podem ser obtidas usando espalhamento.

Os meios de transmissão estão localizados abaixo da camada física. Um meio guiado fornece um condutor físico de um dispositivo para outro. Cabos de par trançado, coaxiais e de fibra óptica são os tipos de meios guiados mais populares. Os meios não guiados (espaço aberto) transportam ondas eletromagnéticas sem usar um condutor físico.

7.7 ATIVIDADES PRÁTICAS

7.7.1 Testes

Diversos testes interativos relativos a este capítulo podem ser encontrados no *site* www.mhhe.com/forouzan. Recomendamos que o estudante realize os testes para verificar o seu entendimento do material apresentado antes de continuar com as atividades práticas.

Questões

7-1 Como podemos determinar o período de uma onda senoidal quando sua frequência é fornecida?

7-2 Qual das seguintes grandezas mede o valor de um sinal a qualquer instante?

a. amplitude **b.** frequência
c. fase

7-3 É possível dizer se um sinal é periódico ou aperiódico apenas analisando sua representação no domínio do tempo? Como?

7-4 A representação no domínio da frequência de um sinal de voz é discreta ou contínua?

7-5 Quais das seguintes opções são causas de problemas de transmissão?

a. atenuação

b. modulação

c. ruído

7-6 Qual das seguintes descrições caracteriza um canal passa-baixas?

a. Um canal cuja largura de banda começa no zero.

b. Um canal cuja largura de banda não começa no zero.

7-7 Enviamos um sinal de voz de um microfone para um gravador. É uma transmissão em banda base ou em banda larga?

7-8 Enviamos um sinal digital de uma estação em uma LAN para outra estação. É uma transmissão em banda base ou em banda larga?

7-9 Qual das seguintes opções corresponde à definição de uma transmissão em banda base?

a. Envio de um sinal digital ou analógico sem modulação usando um canal passa-baixas.

b. Modulação de um sinal digital ou analógico usando um canal passa-faixa.

7-10 Como um sinal periódico composto pode ser decomposto em suas frequências individuais?

7-11 Qual das seguintes opções define a taxa de *bits* máxima teórica para um canal sem ruído?

a. Teorema de Nyquist

b. Capacidade de Shannon

7-12 Qual das seguintes técnicas é um exemplo de mecanismo de conversão digital-digital?

a. codificação de linha

b. codificação de bloco

c. modulação de amplitude

7-13 Defina codificação de bloco e diga qual a sua finalidade.

7-14 Descreva o método PCM.

7-15 Defina transmissão analógica.

7-16 Quais características de um sinal analógico são alteradas para representar o sinal digital em cada um dos seguintes mecanismos de conversão digital-analógico?

a. ASK **b.** PSK

7-17 Qual das quatro técnicas de conversão digital-analógico é mais suscetível a ruído?

a. ASK b. FSK

7-18 Qual componente de um sinal é mostrado no eixo horizontal em um diagrama de constelação?

7-19 Quais características de um sinal analógico são alteradas para representar o sinal analógico passa-baixas em cada uma das seguintes técnicas de conversão analógico-analógico?

a. FM **b.** PM

7-20 Descreva multiplexação.

7-21 Defina o que significa um enlace multiplexado.

7-22 Qual das três técnicas de multiplexação é usada para combinar sinais analógicos?

7-23 Defina TDM síncrona e TDM estatística e compare-as.

7-24 Defina FHSS e explique como essa técnica faz o espalhamento espectral.

7-25 Qual a localização dos meios de transmissão na pilha de protocolos TCP/IP?

7-26 Nomeie as duas principais categorias de meios de transmissão.

7-27 Quais são as três classes principais de meios guiados?

7-28 Para que serve a camada de revestimento em uma fibra óptica?

7-29 Descreva como as ondas viajam da origem até o destino na propagação celeste.

7-30 Descreva como as ondas omnidirecionais são propagadas.

Problemas

7-1 Dadas as frequências listadas a seguir, calcule os períodos correspondentes.

a. 24 Hz **b.** 8 MHz **c.** 140 kHz

7-2 Qual é o deslocamento de fase para cada uma das seguintes ondas?

a. Uma onda senoidal que atinge a amplitude máxima no início do ciclo (tempo igual a zero).

b. Uma onda senoidal que atinge a amplitude máxima após 1/4 de ciclo.

c. Uma onda senoidal que atinge amplitude zero após 3/4 de ciclo e cuja amplitude aumenta em seguida.

7-3 Qual a largura de banda de um sinal que pode ser decomposto em cinco ondas senoidais com frequências de 0, 20, 50, 100 e 200 Hz? Todas as amplitudes de pico são iguais. Desenhe os sinais no domínio da frequência e mostre a largura de banda correspondente.

7-4 Um sinal periódico composto com uma largura de banda de 2.000 Hz é formado por duas ondas senoidais. A primeira tem uma frequência de 100 Hz com uma amplitude máxima de 20 V; a segunda tem uma amplitude máxima de 5 V. Desenhe os sinais e mostre a largura de banda.

7-5 Qual sinal tem uma largura de banda maior: uma onda senoidal com frequência de 100 Hz ou uma onda senoidal com frequência de 200 Hz?

7-6 Qual é a taxa de *bits* para cada um dos seguintes sinais?

a. Um sinal no qual um *bit* dura 0,001 s

b. Um sinal no qual um *bit* dura 2 ms

c. Um sinal no qual 10 *bits* duram 20 μs

7-7 Um dispositivo está enviando dados a uma taxa de 1000 bps.

a. Quanto tempo demora para enviar 10 *bits*?

b. Quanto tempo demora para enviar um caractere (8 *bits*)?

c. Quanto tempo demora para enviar um arquivo de 100.000 caracteres?

7-8 Qual é a taxa de *bits* para o sinal na Figura 7.65?

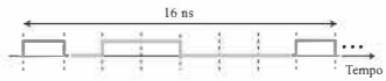



Figura 7.65 Esquema para o Problema 7-8.

7-9 Qual é a frequência do sinal na Figura 7.66?

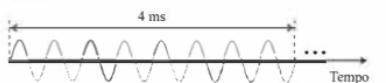



Figura 7.66 Esquema para o Problema 7-9.

7-10 Qual é a largura de banda do sinal composto mostrado na Figura 7.67?

Figura 7.67 Esquema para o Problema 7-10.

7-11 Um sinal periódico composto contém frequências de 10 a 30 kHz, cada uma com uma amplitude de 10 V. Desenhe o espectro de frequências e calcule a largura de banda.

7-12 Um sinal composto aperiódico contém frequências de 10 a 30 kHz. A amplitude de pico é de 10 V para os sinais de frequência mais baixa e mais alta, e de 30 V para o sinal de 20 kHz. Considerando que as amplitudes variem gradualmente da mínima até a máxima e da máxima até a mínima, desenhe o espectro de frequências.

7-13 Um sinal viaja do ponto A ao ponto B. No ponto A, a potência do sinal é de 100 W. No ponto B, a potência é de 90 W. Qual é a atenuação em decibéis?

7-14 A atenuação de um sinal é de –10 dB. Qual é a potência final do sinal se a potência era originalmente de 5 W?

7-15 Um sinal passa por três amplificadores em cascata, cada um com um ganho de 4 dB. Qual é o ganho total? O quanto o sinal foi amplificado?

7-16 Se a largura de banda de um canal é de 5 Kbps, quanto tempo leva para enviar um quadro de 100.000 *bits* por esse canal?

7-17 Um sinal tem um comprimento de onda de 1 μm no ar. Qual é a distância que a frente da onda é capaz de viajar durante 1.000 ciclos?

7-18 Uma linha tem uma relação sinal/ruído de 1.000 e uma largura de banda de 4 kHz. Qual é a taxa de transferência de dados máxima suportada por essa linha?

7-19 Medimos o desempenho de uma linha telefônica (4 kHz de largura de banda). Quando o sinal é de 10 V, o ruído é 10 mV. Qual é a máxima taxa de transferência de dados suportada por essa linha telefônica?

7-20 Um arquivo contém 2 milhões de *bytes*. Quanto tempo leva para transferir esse arquivo usando um canal de 56 Kbps (modem discado)?

7-21 Um monitor de computador apresenta uma resolução de 1.200 × 1.000 *pixels*. Se cada

pixel usa 1.024 níveis de cor, quantos *bits* são necessários para enviar o conteúdo completo de uma tela? Quanto tempo leva para transferir esses dados usando um canal de 56 Kbps (modem discado)?

7-22 Um sinal com 200 miliwatts de energia atravessa 10 dispositivos, cada um com um ruído médio de 2 microwatts. Qual o valor da S/R? Qual o valor da S/R_{dB}?

7-23 Se o valor da tensão de pico de um sinal é 20 vezes o valor da tensão de pico do ruído, qual é o valor da S/R? Qual é o valor da S/R_{dB}?

7-24 Queremos expandir a largura de banda de um canal. Responda às seguintes perguntas:

- **a.** O quanto a taxa de transferência (capacidade do canal) é melhorada se dobrarmos a largura de banda?
- **b.** O quanto a taxa de transferência (capacidade do canal) é melhorada se dobrarmos a S/R?

7-25 Qual é o tempo de transmissão de um pacote enviado por uma estação se o tamanho do pacote for 1 milhão de *bytes* (1 *byte* = 8 *bits*) e a largura de banda do canal for 200 Kbps?

7-26 Qual é o comprimento de um *bit* em um canal se a velocidade de propagação no meio for de 2×10^8 m/s e a largura de banda do canal for de

a. 1 Mbps? **b.** 10 Mbps?

7-27 Quantos *bits* podem existir simultaneamente em um enlace com um atraso de 2 ms se a largura de banda do enlace for de

a. 1 Mbps? **b.** 10 Mbps?

7-28 Qual é o atraso total (latência) para um quadro de 5 milhões de *bits* que está sendo enviado por um enlace com 10 roteadores, cada um apresentando um atraso de fila 2 μs e um tempo de processamento de 1 μs. O comprimento do enlace é de 2.000 km. A velocidade de propagação no enlace é de 210^8 m/s. O enlace apresenta uma largura de banda de 5 Mbps. Qual componente do atraso total é dominante? Qual deles é insignificante?

7-29 Em uma transmissão digital, o relógio do emissor é 0,2 mais rápido do que o relógio do receptor. Quantos *bits* extras por segundo o emissor envia se a taxa de transferência de dados for de 1 Mbps?

7-30 Desenhe o gráfico do esquema NRZ-L usando cada uma das sequências de dados a seguir, considerando que o nível do último sinal foi positivo. A partir dos gráficos, determine a largura de banda desse esquema.

- **a.** 00000000
- **b.** 11111111
- **c.** 01010101
- **d.** 00110011

7-31 Desenhe o gráfico do esquema Manchester usando cada uma das sequências de dados a seguir, considerando que o nível do último sinal foi positivo. A partir dos gráficos, determine a largura de banda desse esquema.

- **a.** 00000000
- **b.** 11111111
- **c.** 01010101
- **d.** 00110011

7-32 Quantas sequências de código inválidas (não utilizadas) podem existir na codificação 5B/6B?

7-33 Qual o resultado do embaralhamento (*scrambling*) da sequência 11100000000000 usando a técnica de codificação B8ZS? Suponha que o nível do último não zero tenha sido positivo.

7-34 Qual é a taxa de amostragem de Nyquist para cada um dos seguintes sinais?

- **a.** Um sinal passa-baixas com largura de banda de 200 kHz.
- **b.** Um sinal passa-faixa com largura de banda de 200 kHz e cuja frequência mais baixa é de 100 kHz.

7-35 Amostramos um sinal passa-baixas com largura de banda de 200 kHz usando 1024 níveis de quantização.

- **a.** Calcule a taxa de *bits* do sinal digitalizado.
- **b.** Calcule a S/R_{dB} desse sinal.
- **c.** Calcule a largura de banda PCM desse sinal.

7-36 Qual é a taxa máxima de transferência de dados de um canal cuja largura de banda é de 200 kHz se usarmos quatro níveis para representar o sinal digital?

7-37 Um sinal analógico apresenta uma largura de banda de 4 kHz. Se amostrarmos esse sinal e o enviarmos por um canal de 30 Kbps, qual será o valor da S/R_{dB}?

7-38 Temos um canal de banda base com uma largura de banda de 1 MHz. Qual é a taxa de transferência de dados desse canal se usarmos cada um dos seguintes esquemas de codificação de linha?

a. NRZ-L

b. Manchester

7-39 Calcule a taxa de *bauds* referente às seguintes taxas de *bits* e tipos de modulação.

a. 2.000 bps, FSK b. 4.000 bps, ASK

c. 36.000 bps, 64-QAM

7-40 Calcule a taxa de *bits* referente às seguintes taxas de *bauds* e tipos de modulação.

a. 1.000 *baud*, FSK

b. 1.000 *baud*, ASK

c. 1.000 *baud*, 16-QAM

7-41 Qual é o número de *bits* por *baud* nas seguintes técnicas?

a. FSK com 8 frequências

b. QAM com uma constelação de 128 pontos

7-42 Desenhe o diagrama de constelação para as seguintes situações:

a. ASK, com valores de amplitude de pico de 1 e 3

b. 8-QAM com dois valores de amplitude de pico diferentes, 1 e 3, e quatro fases distintas.

7-43 Quantos *bits* por *baud* podemos enviar em cada um dos seguintes casos, nos quais o gráfico de constelação do sinal tem os números de pontos dados a seguir?

a. 2 b. 4 c. 16 d. 1024

7-44 Qual é a largura de banda necessária para os seguintes casos se precisarmos enviar 4.000 bps? Considere $d = 1$.

a. ASK

b. FSK ($2\Delta f = 4$ kHz)

c. 16-QAM

7-45 Uma empresa tem um meio de comunicação com uma largura de banda de 1 MHz (passa-baixas). A empresa precisa criar 10 canais independentes distintos, cada um capaz de enviar dados a taxas de pelo menos 10 Mbps. A empresa decidiu usar a tecnologia QAM. Qual é o número mínimo de *bits* por *baud* para cada canal? Qual é o número de pontos no diagrama de constelação para cada canal? Considere $d = 0$.

7-46 Determine a largura de banda necessária para modular um sinal de voz de 5 kHz usando cada uma das seguintes técnicas.

a. AM b. FM ($\beta = 5$) c. PM ($\beta = 1$)

7-47 Para exemplificar o teorema de Nyquist, amostraremos uma onda senoidal simples usando três taxas de amostragem: $f_s = 4f$ (duas vezes a taxa de Nyquist), $f_s = 2f$ (taxa de Nyquist), e $f_s = f$ (metade da taxa de Nyquist). Mostre como podemos reconstruir a onda.

7-48 Suponha que um canal de voz ocupe uma largura de banda de 4 kHz. Precisamos multiplexar 10 canais de voz com bandas de guarda de 500 Hz usando FDM. Calcule a largura de banda necessária.

7-49 Precisamos transmitir 100 canais de voz digitalizada usando um canal passa-faixa de 20 kHz. Qual deve ser a razão de *bits*/Hz se não usarmos bandas de guarda?

7-50 Precisamos usar TDM síncrona e combinar os sinais digitais de 20 fontes, cada um de 100 Kbps. Cada parcela de tempo de saída transporta um *bit* proveniente do sinal digital de cada fonte, mas um *bit* extra é adicionado a cada quadro para a sincronização. Responda às seguintes perguntas:

a. Qual é o tamanho de um quadro de saída, em *bits*?

b. Qual é a taxa de saída de quadros?

c. Qual é a duração de um quadro de saída?

d. Qual é a taxa de saída de dados?

e. Qual é a eficiência do sistema (razão entre *bits* úteis e *bits* totais)?

7-51 Temos 14 fontes, cada uma criando 500 caracteres de 8 *bits* por segundo. Como apenas algumas dessas fontes encontram-se ativas a qualquer momento, usamos o esquema de TDM estatística para combinar os sinais dessas fontes usando caracteres intercalados. Cada quadro requer 6 parcelas de tempo de cada vez, mas precisamos adicionar 4 *bits* de endereços em cada parcela de tempo. Responda às seguintes perguntas:

a. Qual é o tamanho de um quadro de saída em *bits*?

b. Qual é a taxa de saída de quadros?

c. Qual é a duração de um quadro de saída?

d. Qual é a taxa de saída de dados?

7-52 Mostre o conteúdo dos cinco quadros de saída para um multiplexador TDM síncrono que combina os sinais de quatro fontes enviando os caracteres a seguir. Perceba que os caracteres são enviados na mesma ordem em que são digitados. A terceira fonte permanece em silêncio.

a. Mensagem da fonte 1: OLA

b. Mensagem da fonte 2: OI

c. Mensagem da fonte 3:

d. Mensagem da fonte 4: TCHAU

7-53 A Figura 7.68 mostra uma TDM síncrona. Cada quadro de saída tem apenas 10 *bits* de comprimento (3 *bits* vindos de cada entrada mais 1 *bit* de enquadramento). Qual é a sequência de dados na saída?

Figura 7.68 Esquema para o Problema 7-53.

7-54 Um sistema de FHSS utiliza uma sequência de ruído pseudoaleatório (RP) de 4 *bits*. Supondo uma taxa de *bits* de RP de 64 *bits* por segundo, responda às seguintes perguntas:

a. Qual é o número total de saltos possíveis?

b. Qual é o tempo necessário para completar um ciclo completo de RP?

7-55 Temos um meio digital com uma taxa de transferência de dados de 10 Mbps. Quantos canais de voz de 64 kbps podem ser transportados por esse meio se usarmos DSSS com a sequência de Barker?

7-56 Considere que a potência de um sinal sem ruído seja de 10 mW. O sinal passa por cinco estágios de um amplificador. Cada estágio duplica a potência, mas também acrescenta 10 μW de ruído ao sinal. Qual é a S/R_{dB} do sinal depois de deixar o último estágio?

7-57 Qual é o valor da S/R_{dB} quando quantizamos uma voz humana usando 8 *bits* por amostra?

7-58 Temos um canal de banda base de 5 kHz. Precisamos enviar dados a uma taxa de transferência de 40 Kbps utilizando esse canal.

a. Qual deve ser o número de níveis de *bit* para obter essa taxa?

b. Qual deve ser a qualidade mínima do canal (S/R) para obter essa taxa?

7-59 Um cabo STP apresenta uma perda de 1 dB por km a 10 kHz. Precisamos criar um enlace de 10 km usando esse cabo. Qual deve ser a potência do sinal na origem se desejarmos que o sinal apresente 10 mW de potência no destino?

CAPÍTULO

8 MULTIMÍDIA E QUALIDADE DE SERVIÇO

O termo multimídia refere-se à integração de diversos tipos de mídias distintos, como texto, imagens, áudio e vídeo, que são gerados, armazenados e transmitidos digitalmente e podem ser acessados de forma interativa. Atualmente, multimídia é um tema amplo que não pode ser discutido em apenas um capítulo. Neste capítulo, apresentamos uma visão geral da área de multimídia, abordando assuntos que são, direta ou indiretamente, relacionados ao tema de multimídia, como compressão de dados e qualidade de serviço. Dividimos este capítulo em cinco seções:

- Na primeira seção, discutimos a ideia geral por trás da compressão de dados. Embora a compressão não esteja diretamente relacionada ao tema de multimídia, a transmissão multimídia não é possível sem que os dados sejam antes comprimidos.
- Na segunda seção, discutimos os elementos de multimídia: texto, imagem, vídeo e áudio. Mostramos como esses elementos são representados, codificados e comprimidos usando as técnicas discutidas na primeira seção.
- Na terceira seção, separamos a multimídia na Internet em três categorias: transmissão em fluxo contínuo (*streaming*) de áudio/vídeo armazenado, transmissão em fluxo contínuo de áudio/vídeo ao vivo e áudio/vídeo interativo em tempo real. Descrevemos brevemente os recursos e as características de cada uma dessas categorias e apresentamos alguns exemplos.
- Na quarta seção, nos concentramos na categoria interativa em tempo real. Apresentamos dois protocolos que são usados nessa categoria para propósitos de sinalização: SIP e H.323. Eles são usados em soluções de voz sobre IP (telefonia via Internet) e podem ser empregados em protocolos de sinalização de aplicações futuras. Discutimos também protocolos da camada de transporte utilizados por aplicações multimídia. Mostramos que eles (UDP e TCP) não são exatamente adequados para multimídia. Apresentamos um novo protocolo da camada de transporte, o RTP, e o seu componente de controle, o RTCP. Discutimos também o novo protocolo da camada de transporte, denominado SCTP, conforme prometemos no Capítulo 3.
- Na quinta seção, abordamos a qualidade do serviço (QoS – Quality of Service), algo mais necessário em comunicações multimídia do que em comunicações envolvendo apenas texto. Essa seção é apenas uma introdução para um assunto muito interessante e controverso.

8.1 COMPRESSÃO

Nesta seção, discutimos sobre compressão de dados, algo que desempenha um papel crucial na comunicação multimídia devido ao grande volume de dados trafegados. Por meio da compressão, reduzimos o volume de dados a serem transmitidos. Podemos dividir a compressão em duas grandes categorias: compressão com perdas e sem perdas. Analisamos brevemente os métodos geralmente utilizados em cada categoria. Esta seção pode ser ignorada se o leitor estiver familiarizado com técnicas de compressão; nós a incluímos para fornecer a base necessária para os leitores que não estão familiarizados com essa técnica.

8.1.1 Compressão sem perdas

Na **compressão sem perdas**, a integridade dos dados é preservada, pois os algoritmos de compressão e descompressão são inversos exatos um do outro: nenhuma parte dos dados é perdida no processo. Métodos de compressão sem perdas são normalmente utilizados quando não podemos nos dar ao luxo de perder informações. Por exemplo, não pode haver perda de informação quando comprimimos um arquivo de texto ou um aplicativo. A compressão sem perdas também é aplicada como a etapa final em alguns mecanismos de compressão com perdas, com o objetivo de reduzir ainda mais o tamanho dos dados.

Discutimos quatro métodos de compressão sem perdas nesta seção: codificação RLE, codificação baseada em dicionário, codificação de Huffman e codificação aritmética.

Codificação RLE

A **Codificação por Carreira** (codificação RLE – Run-Length Encoding) é o método mais simples para remover redundâncias. Ela pode ser usada para comprimir dados compostos por quaisquer combinações de símbolos. O método substitui uma sequência repetida do mesmo símbolo, ou *carreira*, por duas entidades: um valor de contagem e o símbolo em si. Para ilustrar, mostramos a seguir como podemos comprimir uma sequência de 17 caracteres usando uma sequência de 10 caracteres.

Uma versão modificada desse método pode ser usada se existirem apenas dois símbolos nos dados, tal como uma sequência binária composta por 0s e 1s. Nesse caso, usamos apenas a contagem de um dos símbolos aparecendo entre cada ocorrência do outro símbolo. A Figura 8.1 mostra uma sequência binária na qual existem mais 0s do que 1s. Simplesmente mostramos os números de 0s que aparecem entre 1s.

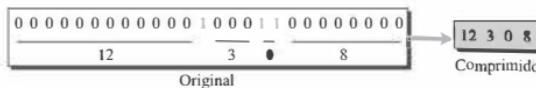



Figura 8.1 Uma versão da codificação RLE para comprimir sequências binárias.

Os dados comprimidos podem ser codificados em binário usando um número fixo de *bits* por dígito. Por exemplo, usando 4 *bits* por dígito, os dados comprimidos podem ser representados como 1100001100001000, de modo que a taxa de compressão é de 26/16 ou quase 1,62 nesse exemplo.

Codificação baseada em dicionário

Existe um grupo de métodos de compressão baseados na criação de um dicionário (vetor) de cadeias de caracteres no texto. A ideia é codificar sequências de caracteres comuns em vez de codificar cada caractere separadamente. O dicionário é criado à medida que a mensagem é lida e, se uma sequência de caracteres que seja uma entrada no dicionário for encontrada na mensagem, o código (índice) daquela entrada é colocado no lugar de tal sequência. A técnica de compressão que discutimos aqui foi inventada por Lempel e Ziv e refinada por Welch. Ela é conhecida como **LZW** (Lempel-Ziv-Welch). O ponto mais interessante sobre essa técnica de codificação é que a criação do dicionário é dinâmica; ele é criado pelo emissor e pelo receptor durante os processos de codificação e decodificação; ele não é enviado pelo emissor para o receptor.

Codificação

O processo de codificação é mostrado a seguir:

1. O dicionário é inicializado com uma entrada para cada caractere possível na mensagem (ou seja, todo o alfabeto). Ao mesmo tempo, um *buffer*, que denotamos por *S*, é inicializado com o primeiro caractere da mensagem. O *buffer S* contém a maior sequência codificável encontrada até o momento. Na etapa de inicialização, apenas o primeiro caractere da mensagem é codificável.
2. O processo examina a mensagem e lê o próximo caractere da mensagem.
   a. Se a concatenação de *S* com o caractere lido estiver no dicionário, o valor atual de *S* não é a maior sequência codificável. O processo atualiza *S*, concatenando o caractere lido ao final dele, e espera pela próxima iteração.
   b. Se a concatenação de *S* com o caractere lido não estiver no dicionário, a maior sequência codificável é o próprio *S*, não o resultado de sua concatenação com o caractere. Três ações são tomadas. Primeiro, o processo adiciona o resultado dessa concatenação ao dicionário, criando uma nova entrada. Segundo, o processo codifica *S*. Terceiro, o processo reinicializa *S*, que passa a ser igual ao caractere lido, para a próxima iteração.
3. O processo repete o passo 2 enquanto houver caracteres na mensagem.

A Tabela 8.1 mostra o pseudocódigo para o processo de codificação. Denotamos o próximo caractere como "caract" e por "S" o *buffer*.

Tabela 8.1 Codificação *LZW*

```
CodificaçãoLZW (mensagem)
{
        Inicializar (Dicionário)
        caract = Ler (primeiro caractere)
        S = caract                                          // S é a sequência codificável
        enquanto (há mais caracteres na mensagem)
        {
                caract = Ler (próximo caractere);
                se ((S + caract) está no Dicionário)        // S não é a sequência codificável
                {
                        S = S + caract;
                }
                senão                                       // S é a sequência codificável
                {
                        adicionarAoDicionário (S + caract);
                        Saída (índice de S no Dicionário);
                        S = caract;
                }
        }
        Saída (índice de S no Dicionário);
}
```

Exemplo 8.1

Mostraremos um exemplo de codificação LZW usando uma mensagem de texto na qual o alfabeto é formado por dois caracteres: A e B (Figura 8.2).

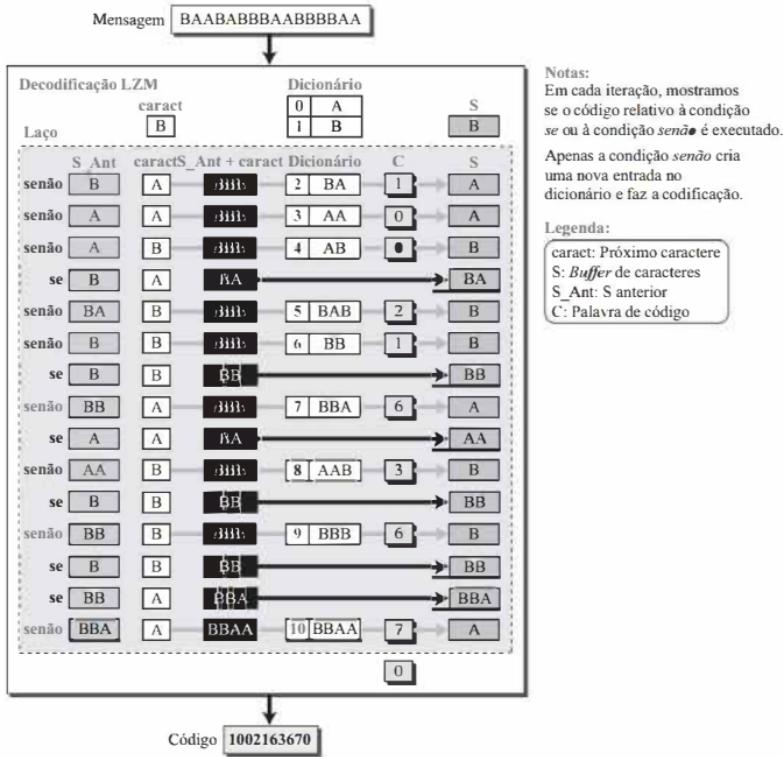



Figura 8.2 Esquema para o Exemplo 8.1.

A Figura 8.2 mostra como o texto "BAABABBBAABBBBAA" é codificado para 1002163670. Observe que o *buffer S_Ant* conserva o valor *S* vindo da iteração anterior antes que ele seja atualizado.

Decodificação

O processo de decodificação é mostrado a seguir:

1. O dicionário é inicializado, conforme explicamos no processo de codificação. A primeira palavra de código é lida e, usando o dicionário, o primeiro caractere na mensagem é colocado na saída.
2. O processo inicializa então um *buffer S* com o valor da palavra de código lida anteriormente. Em seguida, ele lê uma nova palavra de código.
   a. Se a palavra de código estiver no dicionário, o processo adiciona uma nova entrada ao dicionário. O valor dessa entrada é o valor de *S* concatenado com o primeiro caractere

proveniente da entrada do dicionário relativa à nova palavra de código. Ele também fornece como saída a entrada correspondente à nova palavra de código.

b. Se a palavra de código não estiver no dicionário (o que pode acontecer ocasionalmente), o processo concatena o valor de *S* com o primeiro caractere de *S* e armazena o resultado no dicionário. Ele também fornece como saída o resultado da concatenação.

3. O processo repete o passo 2 enquanto houver palavras de código a serem decodificadas.

A Tabela 8.2 mostra o algoritmo simplificado para a decodificação LZW. Denotamos por C a palavra de código e por S o *buffer* de caracteres.

Tabela 8.2 Decodificação *LZW*.

```
DecodificaçãoLZW (código)
{
        Inicializar (Dicionário);
        C = Ler (primeira palavra de código);
        Saida (Dicionário [C]);
        enquanto (há mais palavras de código no código)
        {
                S = Dicionário [C];
                C = Ler (próxima palavra de código);
                se (C está no Dicionário)                    // Caso normal
                {
                        adicionarAoDicionário (S + primeiroSímboloDe (Dicionário [C]));
                        Saida (Dicionário [C]);
                }
                senão                                        // Caso especial
                {
                        adicionarAoDicionário (S + primeiroSímboloDe (S));
                        Saida (S + primeiroSimboloDe (S));
                }
        }
}
```

Exemplo 8.2

Mostraremos como o código do Exemplo 8.1 pode ser decodificado e a mensagem original recuperada (Figura 8.3). A caixa denominada *C_Ant* guarda a palavra de código da iteração anterior, algo que não é necessário no pseudocódigo, mas é necessário aqui para mostrar o processo com mais clareza. Perceba que nesse exemplo temos apenas um caso especial no qual a palavra de código não está no dicionário. A nova entrada do dicionário precisa ser criada a partir do valor do *buffer S* e do primeiro caractere de S. A saída também é igual à nova entrada.

Codificação de Huffman

Quando codificamos dados em sequências binárias, normalmente usamos um número fixo de *bits* para cada símbolo. Para compactar os dados, podemos considerar a frequência de símbolos e a probabilidade de ocorrência de cada um deles na mensagem. A **codificação de Huffman** atribui

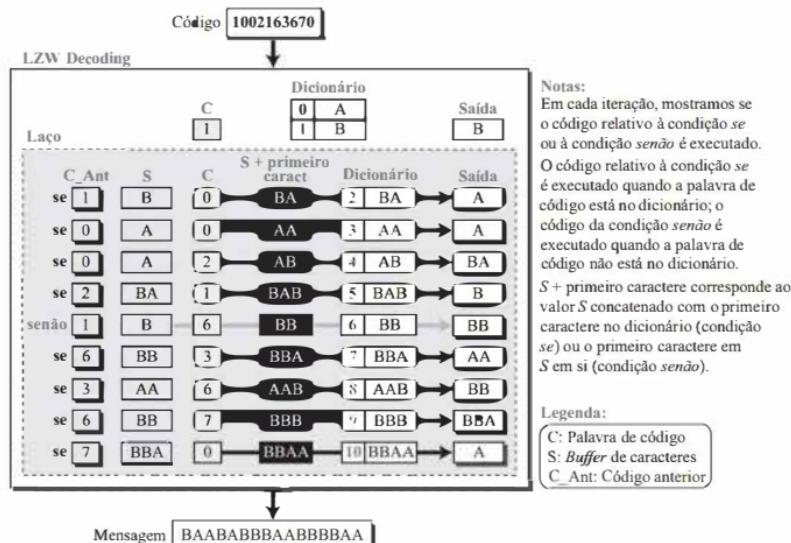



Figura 8.3 Esquema para o Exemplo 8.2.

códigos mais curtos aos símbolos que aparecem mais frequentemente e códigos mais longos para aqueles que aparecem com menor frequência. Por exemplo, imagine que temos um arquivo de texto que usa apenas cinco caracteres (A, B, C, D, E) cujas frequências de ocorrência são iguais a (20, 10, 10, 30, 30).

Árvore de Huffman

Para usar a codificação de Huffman, precisamos primeiramente criar a árvore de Huffman, uma árvore na qual as folhas da árvore são os símbolos. Ela é criada de forma que o símbolo mais frequente seja o mais próximo da raiz da árvore (com o número mínimo de nós até a raiz) e que o símbolo menos frequente seja o mais distante da raiz. A Figura 8.4 ilustra o processo.

1. Colocamos o conjunto completo de caracteres em uma linha. Cada caractere é agora um nó no nível mais baixo da árvore.
2. Selecionamos os dois nós com as frequências mais baixas e os agrupamos para formar um novo nó, resultando em uma árvore simples, de dois níveis. A frequência do novo nó é dada pela soma das frequências dos dois nós originais. Esse nó, um nível acima das folhas, é elegível para ser combinado com outros nós.
3. Repetimos o passo 2 até que todos os nós, em todos os níveis, sejam combinados em uma única árvore.
4. Após a árvore estar pronta, atribuímos valores de *bits* a cada ramo. Como a árvore de Huffman é uma árvore binária, cada nó tem no máximo dois nós-filhos.

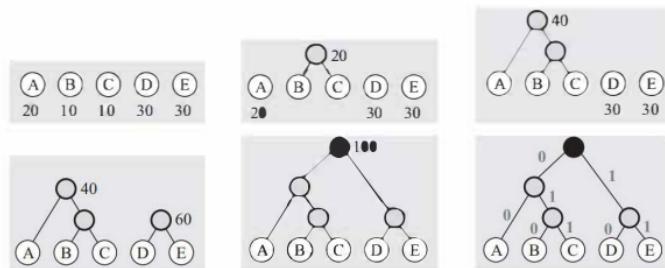



Figura 8.4 Árvore de Huffman.

Tabela de codificação

Após a formação da árvore, podemos criar uma tabela que mostra como cada caractere pode ser codificado e decodificado. O código para cada caractere pode ser determinado começando pela raiz e seguindo os ramos que levam àquele caractere. O código em si corresponde à sequência de *bits* no caminho até o caractere, considerando cada ramo. A Tabela 8.3 mostra os códigos dos caracteres para nosso exemplo simples.

Tabela 8.3 Tabela de codificação.

Símbolo	Código	Símbolo	Código	Símbolo	Código
A	00	C	011	E	11
B	010	D	10		

Observe os seguintes pontos sobre os códigos. Em primeiro lugar, os caracteres com frequências mais altas (A, D e E) recebem um código mais curto do que os caracteres com frequências mais baixas (B e C). Compare essa situação com um código que atribui tamanhos idênticos de *bits* para cada caractere. Em segundo lugar, nesse sistema de codificação, nenhum código é prefixo de outro. Os códigos de 2 *bits*, 00, 10 e 11, não são prefixos de qualquer um dos outros dois códigos (010 e 011). Em outras palavras, não temos um código de 3 *bits* que começa com 00, 10 ou 11. Essa propriedade torna o código de Huffman *instantâneo*. Explicaremos essa propriedade na próxima seção.

Codificação e decodificação

A Figura 8.5 mostra como podemos codificar e decodificar dados usando a codificação de Huffman.

Figura 8.5 Codificação e decodificação usando código de Huffman.

Perceba que conseguimos ter compressão mesmo com uma mensagem pequena. Se quisermos enviar códigos de comprimento fixo usando um alfabeto de cinco caracteres, precisamos de $\log_2 5$ = 2,32 ou 3 *bits* para cada caractere, ou 30 *bits* para a mensagem inteira. Com a codificação de Huffman, precisamos de apenas 22 *bits*. A taxa de compressão é de 30/22 ou 1,36.

Na codificação de Huffman, nenhum código é o prefixo de outro. Isto significa que não precisamos inserir delimitadores para distinguir o código de um caractere do código do caractere seguinte. Essa propriedade da codificação de Huffman também permite a decodificação instantânea: quando o decodificador recebe dois *bits* 00, ele pode imediatamente decodificar esses *bits* para o caractere A; ele não precisa receber mais *bits*.

Um inconveniente da codificação de Huffman é que o codificador e o decodificador devem ambos utilizar a mesma tabela de codificação. Em outras palavras, a árvore de Huffman não pode ser criada dinamicamente como pode ser feito com o dicionário da codificação LZW. No entanto, se o emissor (codificador) e o receptor (decodificador) estiverem usando o mesmo conjunto de símbolos o tempo todo, a árvore pode ser criada e compartilhada uma única vez. Caso contrário, a tabela precisa ser criada pelo emissor e enviada para o receptor.

Codificação aritmética

Nos métodos de compressão descritos anteriormente, cada símbolo ou sequência de símbolos é codificado separadamente. Na **codificação aritmética**, introduzida por Rissanen e Langdon, em 1981, a mensagem inteira é mapeada para um pequeno intervalo entre [0 ; 1). Esse pequeno intervalo é, então, codificado como uma sequência binária. A codificação aritmética é baseada no fato de que podemos ter um número infinito de pequenos intervalos dentro do intervalo semiaberto [0 ; 1). Cada um desses pequenos intervalos pode representar uma das possíveis mensagens que somos capazes de criar usando um conjunto finito de símbolos. A Figura 8.6 ilustra a ideia.

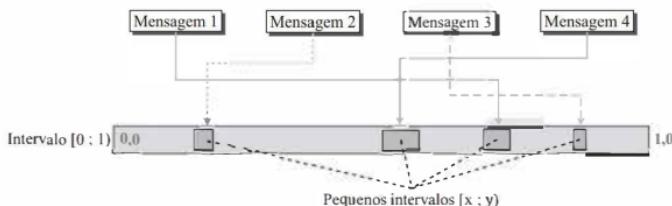



Figura 8.6 Codificação aritmética.

Codificação

Para codificar uma mensagem na codificação aritmética, primeiro precisamos determinar a probabilidade de ocorrência de cada símbolo. Se tivermos M símbolos no alfabeto (incluindo o símbolo de terminação necessário para a decodificação, conforme veremos mais adiante), as probabilidades são $P_1, P_2, ..., P_M$, sendo que $P_1 + P_2 + ... + P_M = 1,0$. A Tabela 8.4 mostra o algoritmo de codificação.

Em cada iteração do laço, divide-se o intervalo atual em M subintervalos, de modo que o comprimento de cada subintervalo seja proporcional à probabilidade do símbolo correspondente. Isto é feito para distribuir uniformemente as mensagens no intervalo [0 ; 1). Também preservamos a ordem dos símbolos no novo intervalo em cada iteração.

A escolha dos *bits* selecionados para a saída depende da implementação. Algumas implementações usam os *bits* que representam a parte fracionária do intervalo inicial.

Tabela 8.4 Codificação aritmética.

```
CodificaçãoAritmética (mensagem)
{
    intervaloAtual = [0 ; 1);
    enquanto (ainda há símbolos na mensagem)
    {
        s = Leitura (próximo símbolo);
        dividir intervaloAtual em subintervalos
        subInt = subintervalo relativo as
        intervaloAtual = subInt
    }
    Escrita (bits relativos ao intervaloAtual)
}
```

Exemplo 8.3

Por razões de simplicidade, consideraremos que nosso conjunto de símbolos é S = {A, B, *}, onde o asterisco corresponde ao símbolo de terminação. Consideramos que a probabilidade de ocorrência de cada símbolo seja

$$P_A = 0{,}4 \qquad P_B = 0{,}5 \qquad P_* = 0{,}1.$$

A Figura 8.7 mostra como podemos determinar o intervalo e o código relativo à mensagem curta "BBAB*".

Inicializamos o intervalo atual em [0 ; 1). Em cada iteração do laço, dividimos o intervalo atual em três subintervalos de acordo com a probabilidade de ocorrência de cada símbolo. Em seguida, lemos o primeiro símbolo e escolhemos o subintervalo correspondente. Atualizamos, então, o intervalo atual para o escolhido. O processo é repetido até que todos os símbolos tenham sido lidos. Após a leitura de cada símbolo, o intervalo atual é reduzido até se tornar [0,685 ; 0,690). Codificamos o limite inferior 0,685 em binário, o que leva a aproximadamente $(0{,}1011)_2$, porém mantemos apenas a parte fracionária, $(1011)_2$, como o valor do código. Observe que quando transformamos um número real entre 0 e 1 para um número binário, podemos acabar obtendo um número infinito de *bits*. Precisamos manter *bits* suficientes para que seja possível recuperar a mensagem original. Manter mais *bits* do que o suficiente não leva a uma codificação eficiente; manter menos *bits* do que o suficiente pode resultar em erros na decodificação. Por exemplo, se usarmos apenas três *bits* (101), temos a representação do valor real 0,625, que está fora do último valor do intervalo atual [0,685 ; 0,690).

Decodificação

A decodificação é semelhante à codificação, mas saímos do laço quando o símbolo de terminação é colocado na saída, por isso precisamos do símbolo de terminação na mensagem original. A Tabela 8.5 mostra o algoritmo de decodificação.

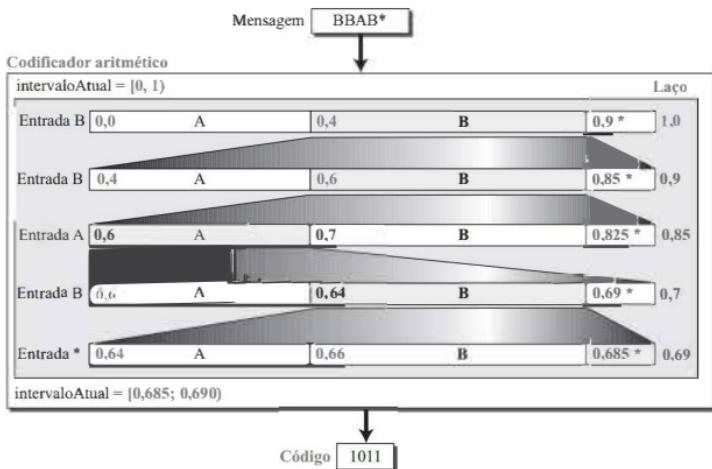



Figura 8.7 Esquema para o Exemplo 8.3.

Tabela 8.5 Decodificação aritmética.

```
DecodificaçãoAritmética (código)
{
        c = Leitura (código)
        num = determinar um número real relativo ao código
        intervaloAtual = [0 ; 1);
        enquanto (verdadeiro)
        {
                dividir intervaloAtual em subintervalos;
                subInt = subintervalo relativo a num;
                Escrita (símbolo relativo a subInt);
                se (símbolo é o símbolo de terminação)    retorna;
                intervaloAtual = subInt;
        }
}
```

Exemplo 8.4

A Figura 8.8 mostra como usamos o processo de decodificação para decodificar a mensagem do Exemplo 8.3. Observe que a mão na figura mostra a posição do número no intervalo correspondente.

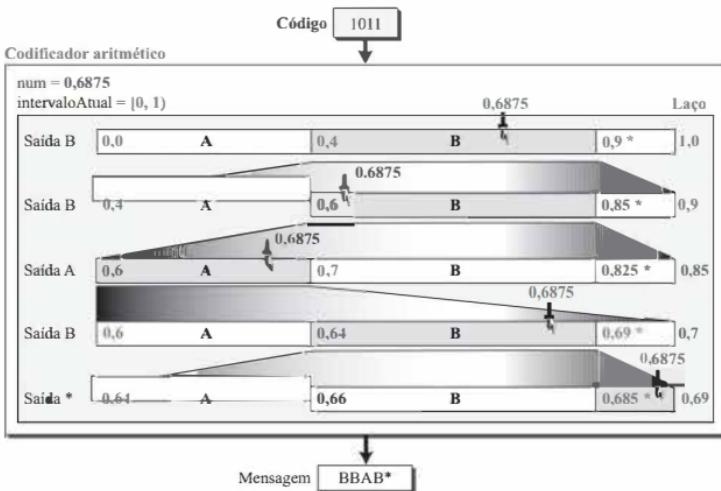



Figura 8.8 Esquema para o Exemplo 8.4.

Codificação aritmética estática *versus* dinâmica

A literatura inclui duas versões de codificação aritmética: *codificação estática* (às vezes chamada *codificação pura*) e *codificação dinâmica* (às vezes chamada *codificação de intervalo*). A versão que discutimos nesta seção é a primeira, a codificação estática. Existem dois problemas com a codificação aritmética estática. Em primeiro lugar, se o intervalo atual for muito pequeno, precisamos usar aritmética de alta precisão para codificar a mensagem, o que resulta em uma grande quantidade de *bits* 0 no meio do código. Em segundo lugar, a mensagem não pode ser codificada até que todos os símbolos tenham sido processados; por isso precisamos de um símbolo de terminação durante a decodificação. A nova versão, a codificação aritmética dinâmica, supera esses dois problemas usando um procedimento que produz pares de *bits* imediatamente após cada símbolo ser lido.

8.1.2 Compressão com perdas

A compressão sem perdas é limitada em relação à quantidade de compressão que pode ser obtida. No entanto, em algumas situações, podemos sacrificar um pouco de precisão para aumentar a taxa de compressão. Embora não possamos nos dar ao luxo de perder informações durante a compressão de texto, podemos fazê-lo quando comprimimos imagens, vídeo e áudio. Por exemplo, a visão humana não consegue detectar algumas pequenas distorções que podem resultar da compressão com perdas de uma imagem. Nesta seção, discutimos algumas ideias por trás da compressão com perdas. Na próxima seção, mostramos como essas ideias podem ser utilizadas na execução da compressão de imagens, vídeo e áudio.

Codificação preditiva

A codificação preditiva é utilizada quando digitalizamos um sinal analógico. No Capítulo 7, discutimos o método de Modulação por Código de Pulsos (PCM – Pulse Code Modulation) como uma

técnica que converte um sinal analógico em um digital, utilizando amostragem. Após a amostragem, cada amostra precisa ser quantizada para criar valores binários. A compressão pode ser obtida no passo de quantização usando *codificação preditiva*.

Na PCM, as amostras são quantizadas separadamente. As amostras vizinhas quantizadas, entretanto, estão intimamente relacionadas e têm valores semelhantes. Na **codificação preditiva**, tiramos proveito dessa semelhança. Em vez de quantizar cada amostra separadamente, as diferenças são quantizadas. As diferenças são menores do que as amostras de fato e, portanto, sua representação requer menos *bits*. Muitos algoritmos são baseados nesse princípio. Começamos com a solução mais simples e em seguida passamos para soluções mais sofisticadas.

Modulação Delta

O método mais simples de codificação preditiva é a chamada **Modulação Delta** (DM – Delta Modulation). Seja x_n a representação do valor da função original no intervalo de amostragem n e y_n o valor reconstruído de x_n. A Figura 8.9 mostra os processos de codificação e decodificação na modulação delta. Na PCM, o emissor quantiza as amostras (x_n) e as envia para o receptor.

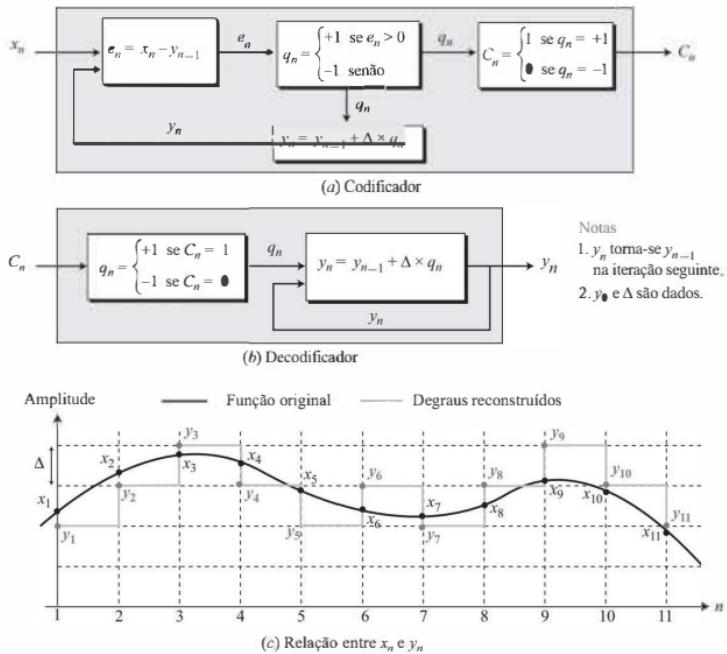



Figura 8.9 Codificação e decodificação na modulação delta.

Na modulação delta, o emissor quantifica e_n, a diferença entre cada amostra (x_n) e o valor reconstruído anterior (y_{n-1}).

O emissor, então, transmite C_n. O receptor reconstrói a amostra y_n a partir do C_n recebido.

Perceba que, para cada amostra, o PCM precisa transmitir vários *bits*. Por exemplo, ele precisa transmitir 3 *bits* para cada amostra se o valor quantizado máximo for 7 (ver Capítulo 7). A técnica de DM reduz o número de *bits* transmitidos porque ela transmite um único *bit* (1 ou 0) para cada amostra.

Podemos nos perguntar o motivo pelo qual a DM quantiza a diferença $x_n - y_{n-1}$ em vez de $x_n - x_{n-1}$. A razão é que a segunda opção faz y variar muito mais rápido do que x caso x seja uma função que varia lentamente. A quantização de $x_n - y_{n-1}$ é autorregulada para um sinal x que apresente crescimento ou queda lentos. A Figura 8.10 compara a reconstrução de um degrau resultante da quantização de $x_n - x_{n-1}$ *versus* $x_n - y_{n-1}$, considerando uma função de crescimento lento. Para uma função de queda lenta, a ideia é similar.

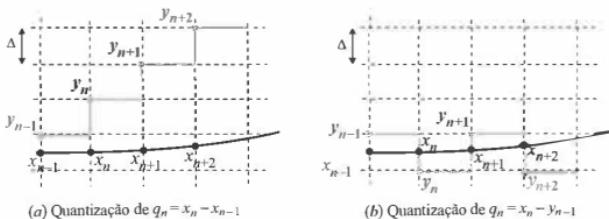



Figura 8.10 Reconstrução resultante da quantização de $x_n - x_{n-1}$ *versus* $x_n - y_{n-1}$.

DM Adaptativa

A Figura 8.11 explica o papel do quantizador na modulação delta. Na região onde Δ é relativamente pequeno em comparação com a inclinação da função original, o sinal em degrau reconstruído não é capaz de acompanhar a função original; o resultado é um erro conhecido como *distorção por excesso de inclinação* (*slope overload distortion*). Por outro lado, na região onde Δ é relativamente grande em comparação com a inclinação da função original, o sinal em degrau reconstruído continua a oscilar com uma grande amplitude ao redor da função original e causa um erro conhecido como *ruído granular*.

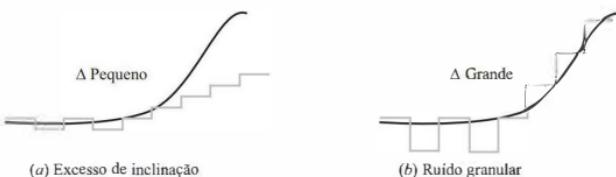



Figura 8.11 Excesso de inclinação e ruído granular.

Como a maioria das funções apresenta regiões com inclinações grandes e pequenas, selecionar um valor grande ou pequeno de Δ diminui um tipo de erro, porém aumenta o outro. A técnica de **DM Adaptativa** (ADM – Adaptive DM) é usada para resolver esse problema. Na ADM, o valor de Δ varia de um passo para outro e é calculado conforme mostrado a seguir.

$$\Delta_n = M_n \Delta_{n-1}$$

onde M_n é denominado *multiplicador do tamanho do degrau* (*step-size multiplier*) e é calculado a partir dos valores de q_n de alguns *bits* precedentes. Existem muitos algoritmos diferentes para determinar o valor de M_n; um algoritmo simples é aumentar M_n de certa porcentagem se q_n permanecer inalterado e reduzi-lo de certa porcentagem caso q_n varie. A adaptação pode ser melhorada ainda mais se atrasarmos o processo de codificação para incluir o conhecimento sobre algumas amostras futuras no cálculo de M_n.

PCM Diferencial

A **PCM Diferencial** (DPCM – Differential PCM) é a generalização da modulação delta. Nela, uma amostra previamente reconstruída y_{n-1} é denominada *preditor* porque é usada para prever o valor atual. Na DPCM, mais do que uma amostra previamente reconstruída é usada para a predição. Nesse caso, a diferença é calculada conforme mostrado a seguir.

$$e_n = x_n - \sum_{i=1}^{N} a_i \, y_{i-1}$$

onde a soma é o preditor, a_i é o *coeficiente do preditor* (ou peso) e N é a *ordem do preditor*. Para a técnica de DM, a ordem do preditor é 1 e $a_1 = 1$. A diferença é quantizada como na técnica DM e o resultado é enviado para o receptor. O receptor reconstrói o valor atual, conforme mostrado a seguir.

$$y_n = \sum_{i=1}^{N} a_i \, y_{i-1} + \Delta q_n$$

Os *coeficientes do preditor* são determinados minimizando-se o erro acumulado entre o valor previsto e o valor real. Uma melhoria é utilizar o *método do erro quadrático*, algo que vai além do escopo deste livro.

DPCM Adaptativa

Pode-se obter uma maior compressão usando coeficientes diferentes para diferentes regiões da amostra, ajustando o quantizador (Δ) de um degrau para outro ou usando essas duas estratégias combinadas. Esse é o princípio por trás da **DPCM Adaptativa** (ADPCM – Adaptive DPCM).

Codificação Preditiva Linear

Na técnica de **Codificação Preditiva Linear** (LPC – Linear Predictive Coding), em vez de enviar sinais de diferenças quantizadas, o emissor analisa os sinais e determina suas características, que incluem frequências na faixa sensível de frequências, potência de cada frequência e duração de cada sinal. Em seguida, o emissor quantiza essa informação e a transmite ao receptor. O receptor transfere a informação para um sintetizador de sinais para simular um sinal semelhante ao original. A LPC pode atingir um nível elevado de compressão. No entanto, esse método é normalmente utilizado pelos militares para comprimir sinais de voz. Nesse caso, a voz sintetizada, embora inteligível, perde a naturalidade e a qualidade necessárias para identificar o interlocutor.

Codificação por transformada

Na codificação por transformada, uma transformação matemática é aplicada ao sinal de entrada para produzir o sinal de saída. A transformação deve ser inversível, de modo a permitir que o sinal original seja recuperado. Ela altera a representação do sinal de um domínio para outro (do domínio do tempo para o domínio da frequência, por exemplo), o que resulta na redução do número de *bits* usados para a codificação.

Precisamos enfatizar que as técnicas baseadas em transformada aplicadas a dados multimídia são, por natureza, sem perdas. No entanto, para atingir os objetivos de compressão, adiciona-se ao processo outra etapa, a quantização, que faz a operação como um todo apresentar perdas.

Transformada Discreta do Cosseno

Uma das transformações populares usadas para dados multimídia é a denominada **Transformada Discreta do Cosseno** (DCT – Discrete Cosine Transform). Apesar de usarmos a técnica de DCT bidimensional na compressão multimídia, discutimos primeiro a DCT unidimensional, que é mais fácil de entender.

DCT unidimensional Na DCT unidimensional, a transformação consiste na multiplicação matricial de uma matriz coluna p (dados de origem) por uma matriz quadrada T (coeficientes da DCT). O resultado é uma matriz coluna M (transformada dos dados). Como a matriz quadrada que representa os coeficientes da DCT é uma matriz ortogonal (inversão e transposição são operações equivalentes), a transformação inversa pode ser obtida multiplicando-se a matriz da transformada dos dados pela transposta da matriz de coeficientes da DCT. A Figura 8.12 mostra o cálculo da transformada na forma de matrizes, onde N é o tamanho da matriz T e T^t representa a matriz transposta de T.

$$\begin{bmatrix} M \end{bmatrix} = \begin{bmatrix} \quad T \quad \end{bmatrix} \times \begin{bmatrix} p \end{bmatrix}$$

(*a*) Cálculo da transformada

$$\begin{bmatrix} p \end{bmatrix} = \begin{bmatrix} \quad T^T \quad \end{bmatrix} \times \begin{bmatrix} M \end{bmatrix}$$

(*b*) Cálculo da transformada inversa

$$T(m, n) = C(m) \cos\left[\frac{\pi n(2m+1)}{2N}\right]$$

para $m = 0$ a $N-1$
para $n = 0$ a $N-1$

$$C(m) = \begin{cases} \sqrt{\frac{1}{N}} & m = 0 \\ \sqrt{\frac{2}{N}} & m > 0 \end{cases}$$

Figura 8.12 DCT unidimensional.

Embora acreditemos que a representação da operação de transformada usando matrizes seja mais fácil de entender, a literatura também usa duas fórmulas para fazê-lo, conforme apresentado na Figura 8.13.

$$\mathbf{M}(m) = \sum_{n=0}^{N-1} C(m) \cos\,[\pi n(2m+1)/(2N)] \times p(n) \qquad \text{para } m = 0, \ldots, N-1$$

$$p(n) = \sum_{m=0}^{N-1} C(n) \cos\,[\pi m(2n+1)/(2N)] \times \mathbf{M}(m) \qquad \text{para } n = 0, \ldots, N-1$$

$$C(i) = \begin{cases} \sqrt{\frac{1}{N}} & i = 0 \\ \sqrt{\frac{2}{N}} & i > 0 \end{cases}$$

Figura 8.13 Fórmulas para as transformadas unidirecionais direta e inversa.

Exemplo 8.5

A Figura 8.14 mostra a matriz usada no cálculo da transformada para N = 4. A primeira linha tem quatro valores iguais, mas as outras linhas têm valores positivos e negativos alternados. Quando cada linha é multiplicada pela matriz dos dados de origem, espera-se que os valores positivos e negativos resultem em valores próximos de zero se os dados de origem tiverem valores próximos uns dos outros. É o que esperamos do cálculo da transformada: que ela mostre que apenas alguns valores nos dados de origem são importantes e que a maioria dos valores é redundante.

$$\underset{\mathbf{M}}{\begin{bmatrix} 203 \\ -2{,}22 \\ 0{,}00 \\ -0{,}16 \end{bmatrix}} = \underset{\mathbf{T}}{\begin{bmatrix} 0{,}50 & 0{,}50 & 0{,}50 & 0{,}50 \\ 0{,}65 & 0{,}27 & -0{,}27 & -0{,}65 \\ 0{,}50 & -0{,}50 & -0{,}50 & 0{,}50 \\ 0{,}27 & -0{,}65 & 0{,}65 & -0{,}27 \end{bmatrix}} \times \underset{p}{\begin{bmatrix} 100 \\ 101 \\ 102 \\ 103 \end{bmatrix}}$$

(*a*) Transformada

$$\underset{p}{\begin{bmatrix} 100 \\ 101 \\ 102 \\ 103 \end{bmatrix}} = \underset{\mathbf{T}^{\mathrm{T}}}{\begin{bmatrix} 0{,}50 & 0{,}65 & 0{,}50 & 0{,}27 \\ 0{,}50 & 0{,}27 & -0{,}50 & -0{,}65 \\ 0{,}50 & -0{,}27 & -0{,}50 & 0{,}65 \\ 0{,}50 & -0{,}65 & 0{,}50 & -0{,}27 \end{bmatrix}} \times \underset{\mathbf{M}}{\begin{bmatrix} 203 \\ -2{,}22 \\ 0{,}00 \\ -0{,}16 \end{bmatrix}}$$

(*b*) Transformada inversa

Figura 8.14 Esquema para o Exemplo 8.5.

O exemplo mostra como podemos transformar a sequência de números (100; 101; 102; 103) em outra sequência (203; −2,22; 0,00; −0,16). Há diversos pontos que queremos mencionar sobre a transformada DCT para explicar melhor as suas propriedades. Em primeiro lugar, a transformação é reversível. Em segundo lugar, a matriz da transformada é ortogonal ($T^{-1} = T^{T}$), o que significa que não é necessário usar a matriz inversa no cálculo da transformada inversa; a matriz transposta pode ser utilizada (cálculo mais rápido). Em terceiro lugar, a primeira linha da matriz M sempre corresponde à média ponderada da matriz *p*. Em quarto lugar, os valores das outras linhas na matriz são muito pequenos (positivos ou negativos), que podem ser ignorados nesse caso. Um ponto muito importante sobre esses três valores é que eles permanecem inalterados se mudarmos os quatro valores na matriz *p*, contanto que a correlação entre esses quatro valores seja mantida. Se alterarmos os dados de origem para *p* = (7, 8, 9, 10), podemos mostrar que os dados após a operação de transformada são M = (17; −2,22; 0,00; −0,16); o primeiro valor mudou porque a média foi alterada, mas o restante dos valores não foi alterado porque a relação entre os valores na matriz de dados não foi modificada. Isso é o que esperamos da operação de transformada, que remove os dados redundantes. Os três últimos valores na matriz *p* são redundantes; eles têm uma relação muito estreita com o primeiro valor.

DCT bidimensional A DCT bidimensional é o que precisamos para comprimir imagens, áudio e vídeo. O princípio é o mesmo, exceto que os dados de origem e a transformada dos dados são matrizes quadradas bidimensionais. Para obter uma transformação com as mesmas propriedades mencionadas para a DCT unidimensional, é preciso usar a matriz T duas vezes (T e T^{T}). O cálculo da transformada inversa também utiliza a matriz T duas vezes, porém na ordem inversa. A Figura 8.15 mostra a DCT bidimensional, na forma de matrizes. A Figura 8.16 mostra a mesma ideia usando duas fórmulas matemáticas.

$$\begin{bmatrix} \mathbf{M} \end{bmatrix} = \begin{bmatrix} \mathbf{T} \end{bmatrix} \times \begin{bmatrix} p \end{bmatrix} \times \begin{bmatrix} \mathbf{T}^{\mathrm{T}} \end{bmatrix}$$

(*a*) Cálculo da transformada

$$\begin{bmatrix} p \end{bmatrix} = \begin{bmatrix} \mathbf{T}^{\mathrm{T}} \end{bmatrix} \times \begin{bmatrix} \mathbf{M} \end{bmatrix} \times \begin{bmatrix} \mathbf{T} \end{bmatrix}$$

(*b*) Cálculo da transformada inversa

$$\mathrm{T}(m, n) = \mathrm{C}(m, n) \cos\left[\frac{m\pi(2n+1)}{2N}\right] \quad \begin{matrix} \text{para } m = 0 \text{ até } N-1 \\ \text{para } n = 0 \text{ até } N-1 \end{matrix}$$

$$\mathrm{C}(m, n) = \begin{cases} \sqrt{\dfrac{1}{N}} & m = 0 \\ \sqrt{\dfrac{2}{N}} & m > 0 \end{cases}$$

Figura 8.15 DCT bidimensional.

$$\mathbf{M}(m, n) = \frac{2}{N}\mathrm{C}(m)\,\mathrm{C}(n)\sum_{k=0}^{N-1}\sum_{l=0}^{N-1} p(k, l)\cos\left[\frac{m\pi(2k+1)}{2N}\right]\cos\left[\frac{n\pi(2l+1)}{2N}\right] \quad \begin{matrix} \text{para } m = 0, \ldots, N-1 \\ \text{para } n = 0, \ldots, N-1 \end{matrix}$$

$$p(k, l) = \frac{2}{N}\mathrm{C}(l)\,\mathrm{C}(k)\sum_{m=0}^{N-1}\sum_{n=0}^{N-1} \mathbf{M}(m, n)\cos\left[\frac{k\pi(2m+1)}{2N}\right]\cos\left[\frac{l\pi(2n+1)}{2N}\right] \quad \begin{matrix} \text{para } k = 0, \ldots, N-1 \\ \text{para } l = 0, \ldots, N-1 \end{matrix}$$

$$\mathrm{C}(u) = \begin{cases} \sqrt{\dfrac{1}{2}} & u = 0 \\ 1 & u > 0 \end{cases}$$

Figura 8.16 Fórmulas para a DCT bidimensional direta e inversa.

8.2 DADOS MULTIMÍDIA

Atualmente, os dados multimídia consistem em *texto*, *imagens*, *áudio* e *vídeo*, embora a definição do termo esteja mudando para incluir tipos futuristas de mídia.

8.2.1 Texto

A Internet armazena uma grande quantidade de texto que pode ser obtido e utilizado. É comum usar o termo texto simples para se referir a uma forma linear de texto, e o termo hipertexto para sua forma não linear. O texto armazenado na Internet usa um conjunto de caracteres, tal como o Unicode, para representar símbolos na linguagem subjacente. Para armazenar uma grande quantidade de dados textuais, o texto pode ser comprimido usando um dos métodos de compressão sem perdas que discutimos anteriormente. Perceba que precisamos usar compressão sem perdas porque não podemos nos dar ao luxo de perder qualquer informação quando executamos a descompressão.

8.2.2 Imagens

No contexto de multimídia, uma imagem (ou uma imagem estática, como é geralmente denominada), é a representação de uma fotografia, de uma página de fax, ou de um quadro em uma imagem em movimento.

Imagem digital

Para que seja possível usar uma imagem, primeiro ela deve ser digitalizada. A *digitalização* nesse caso significa representar uma imagem como uma matriz bidimensional de pontos, denominados *pixels*. Cada *pixel* pode, então, ser representado por um número de *bits*, denominado *profundidade de bits*. Em uma imagem preto e branco, como uma página de fax, a profundidade de *bits* vale 1; cada *pixel* pode ser representado com um *bit* 0 (preto) ou um *bit* 1 (branco). Em uma imagem em tons de cinza, normalmente se usa uma profundidade de *bits* de valor 8, o que resulta em 256 níveis de cor. Em uma imagem colorida, a imagem é normalmente dividida em três canais, sendo que cada canal representa uma das três cores primárias: vermelho, verde ou azul (RGB – Red-Green-Blue). Nesse caso, a profundidade de *bits* vale 24 (8 *bits* para cada cor). Algumas representações usam um canal separado, conhecido como *canal alfa* (α), para representar o plano de fundo. Em uma imagem em preto e branco, isto resulta em dois canais; em uma imagem colorida, resulta em quatro canais.

É óbvio que, à medida que vamos da representação em preto e branco para a representação em tons de cinza e então para imagens em cores, aumenta consideravelmente a quantidade de informação que precisa ser transmitida na Internet. Isto implica na necessidade de comprimir as imagens para economizar tempo de transmissão.

Exemplo 8.6

O exemplo a seguir mostra o tempo necessário para transmitir uma imagem de 1.280 × 720 *pixels* usando uma taxa de transmissão de 100 kbps.

a. Usando uma imagem em preto e branco com uma profundidade de *bits* de 1, precisamos de

Tempo de transmissão = (1.280 × 720 × 1) / 100.000 ≈ 9 segundos

b. Usando uma imagem em tons de cinza com uma profundidade de *bits* de 8, precisamos de

Tempo de transmissão = (1.280 × 720 × 8) / 100.000 ≈ 74 segundos

c. Usando uma imagem colorida com uma profundidade de *bits* de 24, precisamos de

Tempo de transmissão = (1.280 × 720 × 24) / 100.000 ≈ 215 segundos

Compressão de imagens

Embora existam tanto algoritmos de compressão sem perdas quanto com perdas para imagens, nesta seção discutimos o método de compressão com perdas denominado *JPEG*. O padrão **JPEG** (*Joint Photographic Experts Group*, ou Grupo de Especialistas em Fotografia) fornece compressão com perdas, sendo usado na maioria das aplicações. O padrão JPEG pode ser utilizado tanto para imagens coloridas como para imagens em tons de cinza. No entanto, por questões de simplicidade, discutimos apenas imagens em tons de cinza; o método pode ser aplicado a cada um dos três canais de uma imagem colorida. No JPEG, uma imagem em tons de cinza é dividida em blocos de 8 × 8 *pixels*. Os processos de compressão e descompressão envolvem três etapas cada um, conforme mostrado na Figura 8.17.

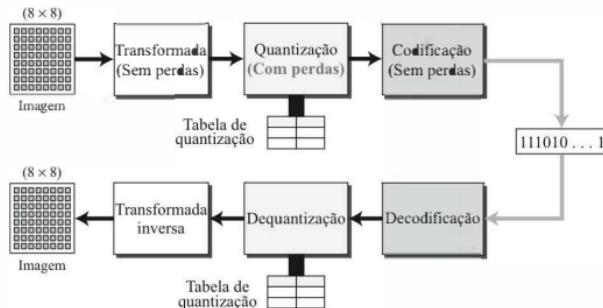



Figura 8.17 Compressão em cada canal do JPEG.

O objetivo de dividir a imagem em blocos é diminuir o número de cálculos, pois, conforme mostramos para a DCT bidimensional, o número de operações matemáticas para cada imagem corresponde ao quadrado do número de unidades presentes.

Transformada

O JPEG normalmente usa a DCT no primeiro passo da compressão e a DCT inversa no último passo. A transformada e a transformada inversa são aplicadas sobre blocos 8 × 8. Discutimos a DCT na seção anterior.

Quantização

A saída da operação de DCT é uma matriz de números reais. A codificação exata desses números reais requer uma grande quantidade de *bits*. O JPEG usa um passo de quantização que não apenas arredonda os valores reais da matriz, mas também transforma alguns valores em zeros, que podem ser eliminados na etapa de codificação para se alcançar uma taxa de compressão mais elevada. Conforme discutido anteriormente, o resultado da operação de DCT define os pesos de diferentes frequências da matriz de origem. Como altas frequências significam mudanças

repentinas no valor de *pixels*, elas podem ser eliminadas porque a visão humana é incapaz de percebê-las. O passo de quantização cria uma nova matriz na qual cada elemento, C(*m*, *n*), é definido conforme mostrado a seguir.

$$C(m, n) = \textbf{arredonda}\ [M(m, n)\ /\ Q(m, n)]$$

onde M(*m*, *n*) é um elemento na matriz transformada e Q(*m*, *n*) é um elemento na matriz de quantização. A função de "arredonda" primeiramente soma 0,5 a um valor real e, em seguida, trunca o valor para um inteiro. Isto significa que 3,7 é arredondado para o inteiro 4, enquanto 3,2 é arredondado para o inteiro 3.

O JPEG define 100 matrizes de quantização, Q1 a Q100, de modo que Q1 fornece a pior qualidade de imagem, porém o mais elevado nível de compressão, enquanto Q100 fornece a melhor qualidade de imagem com o menor nível de compressão. Cabe à aplicação escolher uma dessas matrizes. A Figura 8.18 mostra algumas delas.

80	60	50	80	120	200	255	255
55	60	70	95	130	255	255	255
70	65	80	120	200	255	255	255
70	85	110	145	255	255	255	255
90	110	185	255	255	255	255	255
120	175	255	255	255	255	255	255
245	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255

Q10

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	110	103	99

Q50

3	2	2	3	5	8	10	12
2	2	3	4	5	12	12	11
3	3	3	5	8	11	14	11
3	3	4	6	10	17	16	12
4	4	7	11	14	22	21	15
5	7	11	13	16	12	23	18
10	13	16	17	21	24	24	21
14	18	19	20	22	20	20	20

Q90

Figura 8.18 Três diferentes matrizes de quantização.

Perceba que a única fase do processo que não é completamente reversível é a de quantização. Nesse passo, perdemos algumas informações que não podem ser recuperadas. Na realidade, a única razão pela qual o JPEG é um método de *compressão com perdas* deve-se à fase de quantização.

Codificação

Após a quantização, os valores são reordenados em uma sequência em zigue-zague antes de serem passados para o codificador. Esse reordenamento é feito para permitir que os valores correspondentes às frequências mais baixas sejam passados para o codificador antes dos valores relacionados às frequências mais altas. Como a maioria dos valores de frequência mais elevada é formada por zeros, isto significa que os valores diferentes de zero são passados para o codificador antes dos valores nulos. A Figura 8.19 mostra o processo.

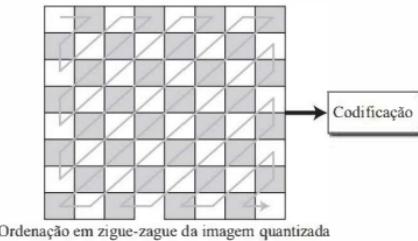



Figura 8.19 Lendo a tabela.

A codificação nesse caso consiste em uma compressão sem perdas usando codificação RLE ou codificação aritmética.

Exemplo 8.7

Para ilustrar a ideia da compressão JPEG, usamos um bloco de imagem em tons de cinza no qual a profundidade de *bits* para cada *pixel* é 20. Usamos um programa Java para calcular a transformada, quantizar e reordenar os valores em uma sequência em zigue-zague; mostramos a codificação na Figura 8.20.

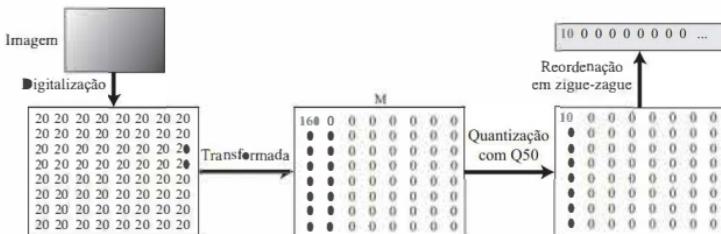



Figura 8.20 Exemplo 8.7: escala de cinza uniforme.

Exemplo 8.8

Como um segundo exemplo, temos um bloco que muda gradualmente; não há mudança brusca entre os valores de *pixels* vizinhos. Ainda obtemos um grande número de valores zero, conforme mostra a Figura 8.21.

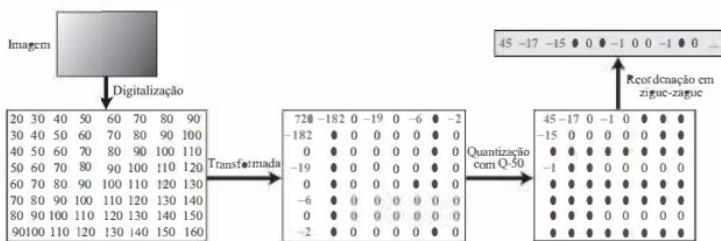



Figura 8.21 Exemplo 8.8: escala de cinza gradual.

Compressão de imagens: GIF

O padrão JPEG usa imagens nas quais cada *pixel* é representado usando 24 *bits* (8 *bits* para cada cor primária). Isto significa que cada *pixel* pode corresponder a uma das 2^{24} (16.777.216) complexas cores possíveis. Por exemplo, um *pixel magenta*, formado pelas componentes vermelho e azul (mas não contém qualquer componente verde), é representado como o número inteiro $(FF00FF)_{16}$.

A maioria das imagens gráficas simples não inclui uma gama tão grande de cores. O formato GIF (*Graphic Interchange Format*, ou Formato para Troca de Gráficos) usa uma pequena paleta (uma tabela indexada) de cores que normalmente contém $2^8 = 256$ cores. Em outras palavras, o GIF mapeia uma cor *real* com uma cor da *paleta*.

Por exemplo, um *pixel* magenta pode ser representado como o número inteiro $(E2)_{16}$ se o magenta for a 226ª cor na paleta. Isto significa que o GIF reduz o tamanho da imagem por um fator de 3 em comparação com o JPEG.

Após a criação da paleta para uma imagem em particular, cada *pixel* pode ser representado por um dentre 256 símbolos (por exemplo, a representação em hexadecimal, usando dois dígitos, correspondente ao índice da cor na paleta). Agora, podemos usar um dos métodos de compressão sem perdas, como codificação baseada em dicionário ou codificação aritmética, para comprimir ainda mais a imagem.

8.2.3 Vídeo

Vídeos são compostos por diversos quadros; cada quadro é uma imagem. Isto significa que um arquivo de vídeo requer uma elevada taxa de transmissão.

Digitalização de vídeo

Um vídeo consiste em uma sequência de quadros. Se os quadros forem apresentados na tela com uma velocidade suficientemente alta, temos uma impressão de movimento. A razão é que nossos olhos não conseguem reconhecer como imagens individuais os quadros aparecendo rapidamente. Não há um número padrão de quadros por segundo; na América do Norte, o uso de 25 frames por segundo é comum. No entanto, para evitar uma condição conhecida como *flickering**, os quadros precisam ser atualizados repetidamente. A indústria de TV redesenha cada quadro duas vezes. Isto significa que 50 quadros precisam ser enviados ou, se houver memória no lado do receptor, são enviados 25 quadros e cada quadro é redesenhado a partir da memória.

Exemplo 8.9

Mostraremos a taxa de transmissão para alguns padrões de vídeo:

a. A transmissão de TV em cores utiliza 720 × 480 *pixels* por quadro, 30 quadros por segundo e 24 *bits* por cor. A taxa de transmissão sem compressão é mostrada a seguir.

$$720 \times 480 \times 30 \times 24 = 248.832.000 \text{ bps} = 249 \text{ Mbps}$$

b. A transmissão de TV em cores de alta definição utiliza 1920 × 1080 *pixels* por quadro, 30 quadros por segundo e 24 *bits* por cor. A taxa de transmissão sem compressão é mostrada a seguir.

$$1920 \times 1080 \times 30 \times 24 = 1.492.992.000 \text{ bps} = 1.5 \text{ Gbps}$$

Compressão de vídeo: MPEG

O **MPEG** (*Motion Picture Experts Group*, ou Grupo de Especialistas em Imagens em Movimento) é um método de compressão de vídeo. Em princípio, uma imagem em movimento é formada por um fluxo rápido de um conjunto de quadros, onde cada quadro é uma imagem. Em outras palavras,

* N. de T.: O efeito de *flickering*, também conhecido como cintilação ou tremulação, consiste em flutuações da intensidade luminosa. Para evitar que tais flutuações em telas (por exemplo, de TVs) causem desconforto visual, é importante reforçar repetidamente a imagem projetada, de modo que o esvanecimento natural da imagem não seja percebido pelo observador.

um quadro é uma combinação espacial de *pixels* e um vídeo é uma combinação temporal de quadros enviados um após o outro. Comprimir vídeo, portanto, significa comprimir espacialmente cada quadro e comprimir temporalmente um conjunto de quadros.

Compressão espacial

A **compressão espacial** de cada quadro é feita usando o JPEG (ou uma modificação dele). Cada quadro é uma figura que pode ser comprimida de forma independente.

Compressão temporal

Na **compressão temporal**, os quadros redundantes são removidos. Quando assistimos à televisão, recebemos 50 quadros por segundo. No entanto, a maioria dos quadros consecutivos é quase idêntica. Por exemplo, quando alguém está falando, a maior parte do quadro permanece constante, exceto pela região ao redor dos lábios da pessoa que fala, que se altera de um quadro para outro.

Para comprimir temporalmente os dados, o método MPEG primeiramente divide um conjunto de quadros em três categorias: quadros I, quadros P e quadros B. A Figura 8.22 mostra como um conjunto de quadros (7, na Figura) é comprimido para criar outro conjunto de quadros.

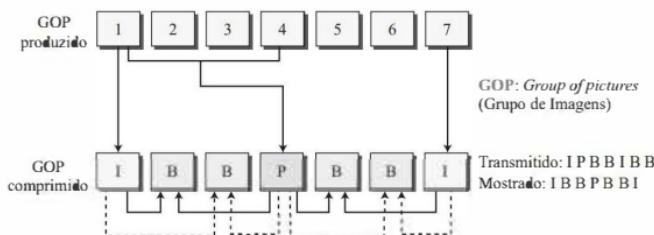



Figura 8.22 Quadros MPEG.

- Um **quadro intracodificado (quadro I)** é um quadro independente que não tem relação com qualquer outro quadro (nem com o quadro enviado anteriormente nem com o enviado em seguida). Eles aparecem em intervalos regulares. Um quadro I deve aparecer periodicamente para tratar alguma mudança repentina no quadro que os quadros anterior e posterior não são capazes de mostrar. Além disso, quando um vídeo é transmitido abertamente, os espectadores podem sintonizar a qualquer momento. Se houver um único quadro I no início da transmissão, o espectador que sintonizar após o envio deste não receberá uma imagem completa. Os quadros I são independentes de outros quadros e não podem ser construídos a partir de outros quadros.
- **Quadros P.** Um **quadro predito (quadro P)** está relacionado com o quadro I ou com o quadro P anterior. Em outras palavras, cada quadro P carrega apenas as alterações relativas ao quadro anterior. Quadros P podem ser construídos usando apenas os quadros I ou P anteriores. Quadros P carregam muito menos informações do que outros tipos de quadro e carregam ainda menos *bits* após a compressão.
- Um **quadro bidirecional (quadro B)** está relacionado com os quadros I ou P anterior e posterior. Em outras palavras, cada quadro B é relativo ao passado e ao futuro. Perceba que um quadro B nunca é relativo a outro quadro B.

8.2.4 Áudio

Sinais de áudio (som) são sinais analógicos que precisam de um meio físico para viajar; eles não podem viajar no vácuo. A velocidade do som no ar é cerca de 330 m/s (1.224 km/h). A faixa de frequência audível pelo ouvido humano normal é aproximadamente de 20Hz a 20kHz, com audibilidade máxima em torno de 3300 Hz.

Digitalização de áudio

Para que a compressão seja possível, os sinais analógicos de áudio são digitalizados usando um conversor analógico-digital. A conversão analógico-digital consiste em dois processos: amostragem e quantização. Um processo de digitalização conhecido como Modulação por Código de Pulsos (PCM – Pulse Code Modulation) foi discutido em detalhes no Capítulo 7. Este processo envolve a amostragem de um sinal analógico, a quantização da amostra e a codificação dos valores quantizados na forma de fluxos de *bits*. O sinal de voz é amostrado a uma taxa de 8.000 amostras por segundo, com 8 *bits* por amostra; o resultado é um sinal digital de 8.000 × 8 = 64 kbps. Uma música é amostrada a 44.100 amostras por segundo, com 16 *bits* por amostra; o resultado é um sinal digital de 44.100 × 16 = 705,6 kbps para áudio do tipo mono e 1.411 Mbps para áudio estéreo.

Compressão de áudio

Os algoritmos de compressão com perdas e sem perdas são ambos usados na compressão de áudio. A compressão de áudio sem perdas nos permite preservar uma cópia exata dos arquivos de áudio; esse método apresenta uma taxa de compressão reduzida, de aproximadamente 2, e é usado principalmente para fins de arquivamento e edição. Algoritmos com perdas fornecem taxas de compressão muito maiores (5 a 20) e são usados em dispositivos de áudio convencionais. Algoritmos com perdas sacrificam um pouco a qualidade, mas reduzem substancialmente os requisitos de espaço em memória e largura de banda. Por exemplo, em um CD, pode-se colocar uma hora de música de alta fidelidade, duas horas de música usando compressão sem perdas ou oito horas de música comprimida por uma técnica de compressão com perdas.

As técnicas de compressão utilizadas para uma conversa e para uma música têm requisitos distintos. As técnicas utilizadas para conversas exigem baixa latência, pois a presença de atrasos significativos acaba degradando a qualidade da comunicação telefônica. Os algoritmos de compressão usados para músicas devem ser capazes de produzir sons de alta qualidade com um número menor de *bits*. Duas categorias de técnicas são usadas na compressão de áudio: *codificação preditiva* e **codificação perceptual**.

Codificação preditiva

Técnicas de codificação preditiva apresentam baixa latência e, portanto, são populares na codificação de fala para conversas telefônicas, nas quais a presença de atrasos significativos pode degradar a qualidade da comunicação. Discutimos vários métodos de codificação preditiva no início deste capítulo: DM, ADM, DPCM, ADPCM e LPC.

Codificação perceptual

Mesmo no melhor dos casos, os métodos de codificação preditiva não são capazes de comprimir suficientemente um sinal de áudio com qualidade de CD para aplicações multimídia. A técnica de compressão mais comum usada para criar áudio com qualidade de CD é a codificação perceptual, baseada na ciência da **psicoacústica**. Os algoritmos usados na codificação perceptual primeiramente transformam os dados no domínio do tempo em dados no domínio da frequência; as operações são, então, feitas sobre os dados no domínio da frequência. Essa técnica, por isso, também é chamada *método do domínio da frequência*.

A psicoacústica consiste no estudo da percepção humana subjetiva do som. A codificação perceptual aproveita-se de falhas no sistema auditivo humano. O limite inferior de audibilidade humana é de 0 dB. Isto somente é verdade para sons com frequências em torno de 2,5 e 5 kHz. O limite inferior é menor para valores de frequência entre essas duas frequências e sobe para frequências fora desse intervalo, conforme mostrado na Figura 8.23*a*. Não somos capazes de ouvir frequências cuja potência fica abaixo dessa curva; logo, não é necessário codificar tais frequências.

Por exemplo, podemos economizar *bits*, sem perda de qualidade, omitindo qualquer som com frequência menor do que 100 Hz se sua potência for inferior a 20 dB.

Podemos economizar ainda mais usando os conceitos de **mascaramento em frequência** e **mascaramento temporal**. O mascaramento em frequência ocorre quando um som alto mascara parcial ou totalmente um som mais baixo se as frequências dos dois estiverem próximas uma da outra. Por exemplo, não conseguimos ouvir nossos amigos em um local onde uma banda de *heavy metal* está tocando muito alto. Na Figura 8.23*b*, um tom de mascaramento elevado, em torno de 700 Hz, aumenta o limiar da curva de audibilidade entre as frequências de cerca de 250 a 1.500 Hz. No mascaramento temporal, um som alto pode anestesiar nosso ouvido por um curto intervalo de tempo, mesmo após o som ter parado.

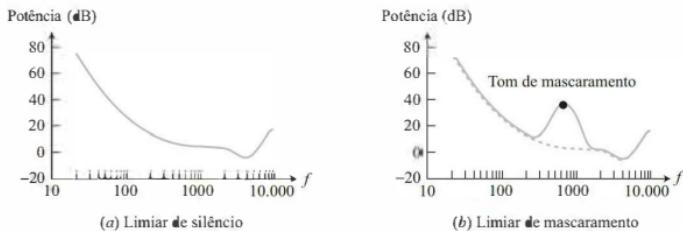



Figura 8.23 Limiar de audibilidade.

A abordagem básica da codificação perceptual é passar a entrada de áudio PCM para duas unidades separadas do codificador, simultaneamente. A primeira unidade é constituída por uma série de filtros de derivação (também chamados *bypass filters*) digitais e é conhecida como *banco de filtros de análise*. Usando uma ferramenta matemática, como uma Transformada Discreta de Fourier (DFT – Discrete Fourier Transform), os filtros dividem a entrada no domínio do tempo em sub-bandas de frequência igualmente espaçadas. Usando a mesma ferramenta matemática ou algo similar, tal como a Transformada Rápida de Fourier (FFT – Fast Fourier Transform), a segunda unidade transforma a entrada do domínio do tempo para o domínio da frequência e determina a frequência de mascaramento para cada sub-banda. Os *bits* disponíveis são, então, alocados de acordo com a propriedade de mascaramento de cada sub-banda: nenhum *bit* é alocado para sub-bandas totalmente mascaradas; um número pequeno de *bits* é alocado para sub-bandas parcialmente mascaradas; e um número grande de *bits* é alocado para sub-bandas não mascaradas. Os *bits* resultantes são codificados novamente para atingir um maior grau de compressão.

MP3

Um padrão que utiliza codificação perceptual é o MP3 (*MPEG Audio Layer 3*, ou Áudio MPEG Camada 3).

8.3 MULTIMÍDIA NA INTERNET

Podemos dividir os serviços de áudio e vídeo em três grandes categorias: *fluxo contínuo de áudio/vídeo armazenado*, *fluxo contínuo de áudio/vídeo ao vivo* e *áudio/vídeo interativo*. O termo *fluxo contínuo*, também conhecido como *streaming*, significa que um usuário pode ouvir (ou ver) o arquivo logo após o início de sua obtenção, mesmo que o arquivo não tenha sido obtido por completo.

8.3.1 Fluxo contínuo de áudio/vídeo armazenado

Na primeira categoria, o fluxo contínuo de áudio/vídeo armazenado, os arquivos são compactados e armazenados em um servidor Um cliente obtém os arquivos por meio da Internet. Isto também é conhecido como *áudio/vídeo sob demanda*. Exemplos de arquivos de áudio armazenados incluem músicas, sinfonias, audiolivros e discursos famosos. Alguns exemplos de arquivos de vídeo armazenados são filmes, programas de TV e videoclipes de música. Podemos dizer que a transmissão contínua de áudio/vídeo armazenado refere-se a solicitações *sob demanda* por arquivos de áudio/vídeo compactados.

Obter esses tipos de arquivos de um servidor Web pode ser diferente de obter outros tipos de arquivos. Para entender o conceito, discutiremos quatro abordagens, cada uma com uma complexidade diferente.

Primeira abordagem: Usando um servidor Web

Um arquivo compactado de áudio/vídeo pode ser obtido na forma de um arquivo de texto. O cliente (navegador Web) pode usar os serviços do HTTP e enviar uma mensagem GET para obter o arquivo. O servidor Web pode enviar o arquivo compactado para o navegador, que, por sua vez, usa um aplicativo auxiliar, normalmente conhecido como um *media player* ou *reprodutor de mídia*, para reproduzir o arquivo. A Figura 8.24 mostra essa abordagem.

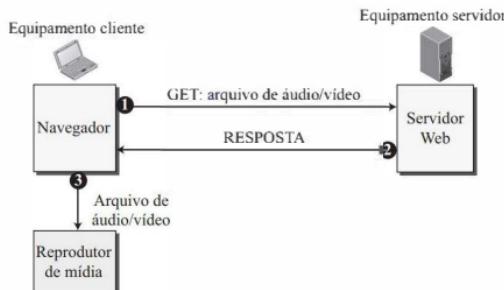



Figura 8.24 Usando um servidor Web.

Essa abordagem é bastante simples e não envolve o conceito de transmissão contínua. No entanto, apresenta uma desvantagem. Um arquivo de áudio/vídeo é normalmente grande, mesmo após compressão. Um arquivo de áudio pode conter dezenas de *megabits*, enquanto um arquivo de vídeo pode conter centenas de *megabits*. Nessa abordagem, o arquivo precisa ser obtido completamente antes de ser reproduzido. Usando taxas de transmissão de dados atuais, são necessários alguns segundos ou décimos de segundo antes que o usuário seja capaz de reproduzir o arquivo.

Segunda abordagem: Usando um servidor Web com um metarquivo

Em uma segunda abordagem, o reprodutor de mídia é conectado diretamente ao servidor Web para obter o arquivo de áudio/vídeo. O servidor Web armazena dois arquivos: o arquivo de áudio/vídeo em si e um **metarquivo** que contém informações sobre o arquivo de áudio/vídeo. A Figura 8.25 mostra a sequência de passos necessários nessa abordagem.

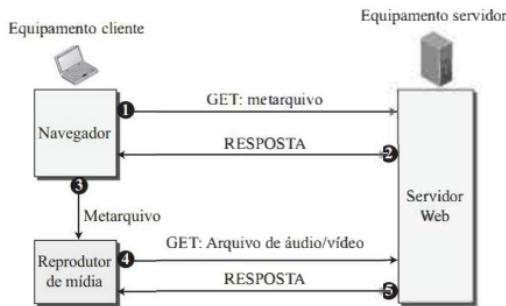



Figura 8.25 Usando um servidor Web com um metarquivo.

1. O cliente HTTP acessa o servidor Web usando uma mensagem do tipo GET.
2. As informações sobre o metarquivo vêm na resposta.
3. O metarquivo é passado para o reprodutor de mídia.
4. O reprodutor de mídia usa o URL no metarquivo para acessar o arquivo de áudio/vídeo.
5. O servidor Web responde.

Terceira abordagem: Usando um servidor de mídia

O problema com a segunda abordagem é que o navegador e o reprodutor de mídia usam ambos os serviços do HTTP, que foi projetado para funcionar sobre TCP. Isto é apropriado para obter o metarquivo, mas não para obter o arquivo de áudio/vídeo. A razão é que o TCP retransmite segmentos perdidos ou danificados, o que vai contra a filosofia de fluxo contínuo. Precisamos dispensar o TCP e seus mecanismos de controle de erro; precisamos usar o UDP. No entanto, o HTTP, usado para acessar o servidor Web, bem como o servidor Web em si, são projetados para usar TCP; precisamos de outro servidor, um **servidor de mídia**. A Figura 8.26 ilustra esse conceito.

1. O cliente HTTP acessa o servidor Web usando uma mensagem de GET.
2. As informações sobre o metarquivo vêm na resposta.
3. O metarquivo é passado para o reprodutor de mídia.
4. O reprodutor de mídia usa o URL no metarquivo para acessar o servidor de mídia e obter o arquivo. A obtenção pode ser feita usando qualquer protocolo baseado em UDP.
5. O servidor de mídia responde.

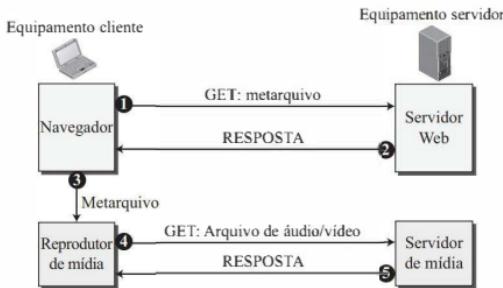



Figura 8.26 Usando um servidor de mídia.

Quarta abordagem: Usando um servidor de mídia e RTSP

O **Protocolo de Fluxo Contínuo em Tempo Real** (RTSP – Real-Time Streaming Protocol) é um protocolo de controle projetado para adicionar funcionalidades extras ao processo de transmissão em fluxo contínuo. Usando o RTSP, podemos controlar a reprodução de áudio/vídeo. O RTSP é um protocolo de controle fora da banda, de modo semelhante à segunda conexão utilizada no FTP. A Figura 8.27 mostra um servidor de mídia e o RTSP.

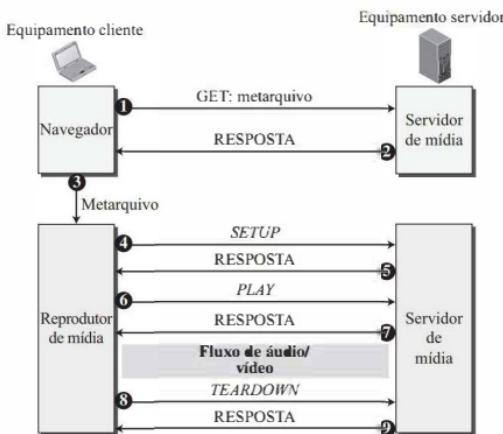



Figura 8.27 Usando um servidor de mídia e RTSP.

1. O cliente HTTP acessa o servidor Web usando uma mensagem de GET.
2. As informações sobre o metarquivo vêm na resposta.
3. O metarquivo é passado para o reprodutor de mídia.

4. O reprodutor de mídia envia uma mensagem de *SETUP* (configuração) para criar uma conexão com o servidor de mídia.
5. O servidor de mídia responde.
6. O reprodutor de mídia envia uma mensagem de *PLAY* (reproduzir) para começar a reprodução (obtenção) da mídia.
7. O arquivo de áudio/vídeo é transmitido usando outro protocolo baseado no UDP.
8. A conexão é finalizada usando a mensagem de *TEARDOWN* (finalização).
9. O servidor de mídia responde.

O reprodutor de mídia pode enviar outros tipos de mensagens. Por exemplo, uma mensagem de *PAUSE* interrompe temporariamente a transmissão, que pode ser retomada com uma mensagem de *PLAY*.

Exemplo: Vídeo Sob Demanda

Serviços de Vídeo Sob Demanda (VOD – Video On Demand) permitem que os usuários selecionem um vídeo dentre um grande número de vídeos disponíveis e assistam a ele de forma interativa: o usuário pode pausar, retroceder, avançar, etc. Um usuário pode assistir ao vídeo em tempo real ou armazená-lo em seu computador, em um reprodutor de mídia portátil ou em um dispositivo como um gravador de vídeo digital (DVR – Digital Video Recorder) e assistir ao conteúdo mais tarde. Provedores de TV a cabo, TV via satélite e IPTV* oferecem serviços de transmissão em fluxo contínuo de VOD tanto para conteúdos gratuitos como do tipo *pay-per-view* (pagamento por visualização). Muitas outras empresas, como a Amazon, e empresas de locação de vídeo, como a Blockbuster, também fornecem serviços de VOD. A Televisão pela Internet é uma forma cada vez mais popular de serviço de vídeo sob demanda.

8.3.2 Fluxo contínuo de áudio/vídeo ao vivo

Na segunda categoria, o fluxo contínuo de áudio/vídeo ao vivo, um usuário ouve uma transmissão de áudio e/ou vídeo via Internet. Bons exemplos desse tipo de aplicação são o rádio e a TV via Internet.

Existem diversas semelhanças entre a transmissão em fluxo contínuo de áudio/vídeo armazenado e a transmissão em fluxo contínuo de áudio/vídeo ao vivo. Ambos são sensíveis a atrasos e nenhum deles aceita a retransmissão de pacotes. Entretanto, existe uma diferença. No primeiro caso, a comunicação é feita via *unicast* e sob demanda. No segundo, a comunicação é feita via *multicast* e ao vivo. Em aplicações de fluxo contínuo ao vivo, é mais adequado usar os serviços de *multicast* do IP e usar protocolos, como UDP e RTP (discutido mais adiante). No entanto, atualmente, aplicações de transmissão em fluxo contínuo ao vivo ainda estão usando TCP e *unicast* múltiplo em vez de *multicast*. Há ainda muito o que se avançar nessa área.

Exemplo: Rádio via Internet

O rádio via Internet, ou rádio Web, é um serviço de transmissão de áudio via Web que oferece notícias, informações esportivas, discussões e música por meio da Internet. Esse serviço envolve uma transmissão acessível a partir de qualquer lugar do mundo. O serviço de rádio Web é fornecido por meio da Internet, mas é semelhante à transmissão de rádio tradicional: o serviço é não interativo e a transmissão não pode ser pausada ou repetida, como no caso de serviços sob demanda. O maior

* N. de T.: Termo IPTV refere-se à transmissão de sinais televisivos usando a pilha de protocolos TCP/IP para realizar a disponibilização do conteúdo. Em português, também pode ser usado o termo TVIP ou TV sobre IP.

grupo de provedores de rádio via Internet atualmente é constituído por estações de rádio existentes que transmitem conteúdo simultaneamente da forma tradicional e via Internet. Ele também inclui estações de rádio que usam exclusivamente a Internet para a transmissão. Em aplicações de rádio Web, o áudio é frequentemente comprimido para o formato MP3 ou similar e os *bits* são transportados usando pacotes TCP ou UDP. Para evitar os efeitos do *jitter* (variação do atraso na rede), no lado do usuário, os *bits* recebidos são colocados no *buffer* (unidades de armazenamento temporário) e sua reprodução é atrasada em alguns segundos, permitindo que eles sejam reagrupados e reproduzidos de forma contínua.

Exemplo: Televisão via Internet

A *Televisão via Internet* (ITV – Internet television) permite que os telespectadores escolham o programa que desejam assistir dentre uma lista de conteúdos. Os principais modelos de televisão via Internet são a transmissão em fluxo contínuo ou vídeo selecionável em um *site*.

Exemplo: IPTV

Televisão sobre Protocolo Internet (IPTV – Internet Protocol Television) é a tecnologia de nova geração utilizada para transmitir conteúdo televisivo em tempo real e interativo. Em vez do sinal de TV ser transmitido via satélite, a cabo ou por vias terrestres, o sinal de IPTV é transmitido pela internet. Perceba que IPTV é diferente de ITV. Uma aplicação de TV via Internet é criada e gerenciada por provedores de serviço que não são capazes de controlar a entrega final dos conteúdos; eles são distribuídos usando a infraestrutura existente da Internet aberta. Serviços de IPTV, por outro lado, são amplamente gerenciados de modo a proporcionar garantias de qualidade de serviço sobre uma rede complexa e cara. A rede usada em sistemas de IPTV é projetada para garantir a entrega eficiente de grandes quantidades de tráfego de vídeo *multicast* e conteúdo HDTV para assinantes.

A plataforma baseada em IP oferece vantagens significativas, incluindo a capacidade de integrar os serviços de televisão com outros serviços baseados em IP, como acesso de alta velocidade à Internet e VoIP. Uma diferença no funcionamento do IPTV com relação à TV a cabo ou via satélite é que em uma rede a cabo ou via satélite típica, todo o conteúdo flui constantemente da estação de TV para cada cliente. O cliente, usando um *set-top box* (um dispositivo que se conecta a um televisor), seleciona o conteúdo. No sistema de IPTV, o conteúdo permanece na rede e apenas o que o cliente seleciona lhe é enviado. A vantagem é que o IPTV requer menos largura de banda e, portanto, permite a entrega de mais conteúdo e funcionalidades. A desvantagem é que a privacidade do cliente pode vir a ser comprometida, pois o provedor de serviços de IPTV poderia monitorar precisamente cada programa assistido por cada cliente.

8.3.3 Áudio/vídeo interativo em tempo real

Na terceira categoria, áudio/vídeo interativo, as pessoas usam a Internet para se comunicarem de forma interativa umas com as outras. O serviço de telefonia via Internet, geralmente denominado **voz sobre IP** (VoIP), é um exemplo desse tipo de aplicação. Uma aplicação de videoconferência é outro exemplo que permite às pessoas se comunicarem visual e oralmente.

Características

Antes de discutirmos os protocolos usados nessa classe de aplicações, discutimos algumas características da comunicação via áudio/vídeo em tempo real.

Relação temporal

Dados em tempo real em uma rede de comutação de pacotes exigem que seja preservada a relação temporal entre pacotes de uma sessão. Por exemplo, considere que um servidor de vídeo em tempo real crie imagens de vídeo ao vivo e as envie pela rede. O vídeo é digitalizado e empacotado. Existem apenas três pacotes, sendo que cada um contém 10 segundos de informações de vídeo. O primeiro pacote começa no instante 00:00:00, o segundo começa no instante 00:00:10 e o terceiro começa no instante 00:00:20. Imagine também que demora 1 s (um exagero, para simplificar o exemplo) para que cada pacote chegue ao destino (atrasos iguais). O receptor pode reproduzir o primeiro pacote no instante 00:00:01, o segundo no instante 00:00:11 e o terceiro no instante 00:00:21. Embora haja uma diferença de tempo de 1 s entre aquilo que o servidor envia e o que o cliente vê na tela do computador, a ação acontece em tempo real. A relação temporal entre os pacotes é preservada. O atraso de 1 s não é importante. A Figura 8.28 ilustra a ideia.

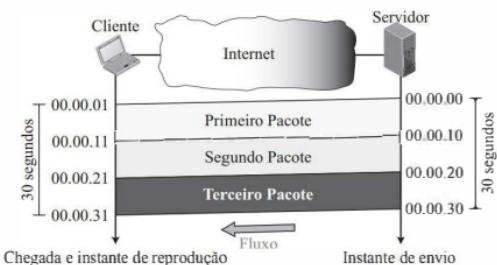



Figura 8.28 Relação temporal.

Mas o que acontece se os pacotes chegarem com atrasos diferentes? Por exemplo, o primeiro pacote chega no instante 0:00:01 (1 s de atraso), o segundo chega no instante 0:00:15 (5 s de atraso) e o terceiro chega no instante 0:00:27 (7 s de atraso). Se o receptor começar a reproduzir o primeiro pacote no instante 00:00:01, ele terminará a reprodução no instante 00:00:11. Entretanto, o próximo pacote ainda não chegou; ele chega 4 s mais tarde. Existe uma lacuna entre o primeiro e o segundo pacotes e entre o segundo e o terceiro pacotes à medida que o vídeo é reproduzido no local remoto. Esse fenômeno é chamado ***jitter***, ou **variação de atraso.** A Figura 8.29 ilustra essa situação.

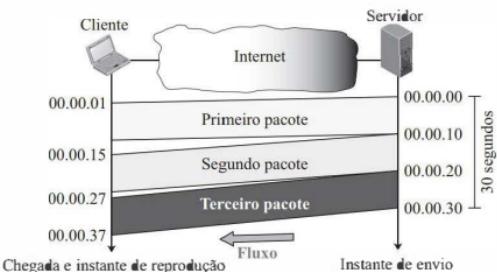



Figura 8.29 *Jitter*

Carimbo de tempo

Uma solução para o *jitter* é usar um **carimbo de tempo**, também conhecido como ***timestamp***. Se cada pacote tiver um carimbo de tempo que mostra o instante em que ele foi gerado com relação ao primeiro pacote (ou ao pacote anterior), então, o receptor pode somar esse tempo ao instante em que ele inicia a reprodução. Em outras palavras, o receptor sabe quando cada pacote deve ser reproduzido. Imagine que o primeiro pacote no exemplo anterior tem um carimbo de tempo de 0, o segundo tem um de 10 e o terceiro, de 20. Se o receptor começar a reproduzir o primeiro pacote no instante 00:00:08, o segundo será reproduzido no instante 00:00:18 e o terceiro no instante 00:00:28. Não há lacunas entre os pacotes. A Figura 8.30 mostra essa situação.

Para evitar o *jitter*, podemos colocar carimbos de tempo (*timestamps*) nos pacotes e desvincular o instante de chegada do instante de reprodução.

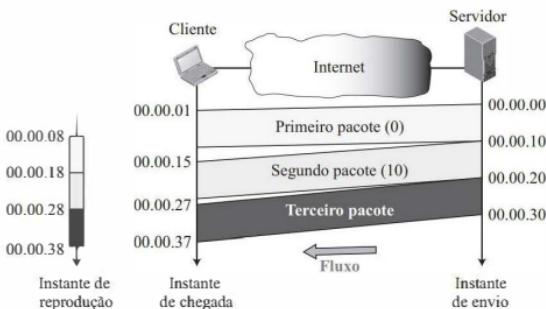



Figura 8.30 Carimbo de tempo (*timestamp*).

Buffer de reprodução

Para que sejamos capazes de desvincular o instante de chegada do instante de reprodução, precisamos de um *buffer* para armazenar os dados até que eles sejam reproduzidos. Tal *buffer* é conhecido como ***buffer* de reprodução**. Quando uma sessão começa (o primeiro *bit* do primeiro pacote chega), o receptor atrasa a reprodução dos dados até um limiar ser atingido. No exemplo anterior, o primeiro *bit* do primeiro pacote chega no instante 00:00:01; o limiar é de 7 s, e o instante de início da reprodução é 00:00:08. O limiar é medido em unidades de tempo dos dados. A reprodução não se inicia até que haja uma quantidade de dados para ocupar um intervalo de tempo igual ao limiar.

Os dados são armazenados no *buffer*, possivelmente a uma taxa variável, porém são extraídos e reproduzidos a uma taxa fixa. Perceba que a quantidade de dados nos *buffers* diminui ou aumenta, mas contanto que o atraso seja menor do que o tempo necessário para reproduzir uma quantidade de dados igual ao limiar, não há *jitter*. A Figura 8.31 mostra o *buffer* em momentos diferentes para o nosso exemplo.

Para entender como um *buffer* de reprodução pode de fato eliminar o *jitter*, precisamos imaginar esse *buffer* como uma ferramenta que introduz mais atraso em cada pacote. Se o atraso adicionado a cada pacote fizer com que o atraso total (na rede e no *buffer*) para cada pacote seja igual, então os pacotes são reproduzidos sem interrupção, como se não houvesse qualquer atraso. A Figura 8.32 ilustra a ideia usando a linha de tempo para sete pacotes. Perceba que precisamos escolher o atraso de *buffer* para o primeiro pacote no *buffer* de tal forma que as duas curvas em dentes de serra mais à direita não se cruzem.

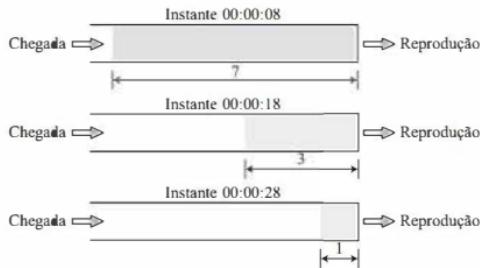



Figura 8.31 *Buffer* de reprodução.

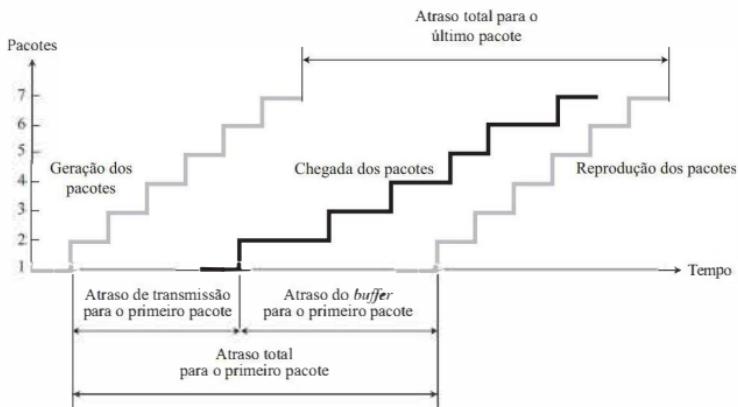



Figura 8.32 A linha de tempo dos pacotes.

Conforme mostra a figura, se o instante de reprodução para o primeiro pacote for selecionado de forma adequada, o atraso total para todos os pacotes deve ser o mesmo. Os pacotes que apresentam um maior atraso de transmissão esperam por menos tempo no *buffer* e vice-versa.

Ordenação

Além de informações sobre a relação temporal e de carimbos de tempo, mais uma característica é necessária no tráfego em tempo real. Precisamos de um *número de sequência* para cada pacote. O carimbo de tempo sozinho não é capaz de informar ao receptor se um pacote for perdido. Por exemplo, considere que os carimbos de tempo sejam 0, 10 e 20. Se o segundo pacote for perdido, o receptor recebe apenas dois pacotes com carimbos de tempo 0 e 20. O receptor considera que o pacote cujo carimbo de tempo é 20 é o segundo, produzido 20 s após o primeiro. O receptor não tem como saber que o segundo pacote foi, na realidade, perdido. Para lidar com essa situação, é necessário um número de sequência para ordenar os pacotes.

Multicast

Dados multimídia desempenham um papel primordial em aplicações de áudio e videoconferência. O tráfego pode ser pesado e os dados são distribuídos usando métodos de *multicast*. Aplicações de conferência exigem uma comunicação bidirecional entre receptores e emissores.

Tradução

Algumas vezes, o tráfego em tempo real requer *tradução*. Um tradutor é um computador capaz de alterar o formato de um sinal de vídeo que usa uma ampla largura de banda em um sinal de menor qualidade e banda mais estreita. Isto é necessário, por exemplo, quando uma fonte cria um sinal de vídeo de alta qualidade a 5 Mbps e o envia para um destinatário cuja largura de banda é inferior a 1 Mbps. Para que esta última receba o sinal, é necessário um tradutor para decodificar o sinal e codificá-lo novamente em uma qualidade mais baixa que exija uma menor largura de banda.

Mixagem

Se houver mais do que uma fonte que pode enviar os dados ao mesmo tempo (como em um vídeo ou em uma audioconferência), o tráfego é composto por múltiplos fluxos. Para convergir o tráfego em um único fluxo, os dados de diferentes fontes podem ser mixados (misturados). Um ***mixer***, ou **misturador**, soma matematicamente sinais provenientes de diferentes fontes para criar um único sinal.

Correção Antecipada de Erros

Discutimos a detecção de erros e a retransmissão no Capítulo 5. Entretanto, a retransmissão de pacotes corrompidos e perdidos não é útil para a transmissão de multimídia em tempo real, porque isto cria um atraso inaceitável na reprodução: teríamos que esperar até que o pacote perdido ou corrompido fosse reenviado. Precisamos corrigir os erros e reproduzir o pacote imediatamente. Diversos esquemas foram concebidos e são utilizados nesse cenário, os quais são conhecidos coletivamente como técnicas de **Correção Antecipada de Erros** (FEC – Forward Error Correction). Aqui, discutimos brevemente algumas das técnicas comuns.

Correção de erros usando a distância de Hamming

No Capítulo 5, discutimos a distância de Hamming para a detecção de erros. Dissemos que, para detectar s erros, a distância de Hamming mínima deve ser $d_{min} = s + 1$. Para a correção de erros, definitivamente precisamos de uma distância maior. Pode ser demonstrado que, para detectar t erros, precisamos ter $d_{min} = 2t + 1$. Em outras palavras, se quisermos corrigir 10 *bits* em um pacote, precisamos fazer com que a distância de Hamming mínima seja de 21 *bits*, o que significa que uma grande quantidade de *bits* redundantes precisa ser enviada com os dados. Para dar um exemplo, considere o famoso código BCH. Nele, se o tamanho dos dados for de 99 *bits*, precisamos enviar 255 *bits* (156 *bits* extras) para corrigir apenas 23 erros de *bits* possíveis. Na maioria das vezes, não podemos dispor de tal redundância. Fornecemos alguns exemplos de como calcular os *bits* necessários na seção de atividades práticas.

Correção de erros usando XOR

Outra recomendação é usar a propriedade da operação de OU exclusivo (XOR) conforme mostrado a seguir.

$$\mathrm{R} = \mathbf{P}_1 \oplus \mathbf{P}_2 \oplus \ldots \oplus \mathrm{P}_i \oplus \ldots \oplus \mathbf{P}_N \quad \rightarrow \quad \mathrm{P}_i \oplus \mathbf{P}_1 \oplus \mathbf{P}_2 \oplus \ldots \oplus \mathrm{R} \oplus \ldots \oplus \mathbf{P}_N$$

Em outras palavras, se aplicarmos a operação de OU exclusivo sobre N itens de dados (P_1 a P_N), podemos recriar qualquer um dos itens calculando o OU exclusivo de todos eles e

substituindo o único item a ser criado pelo resultado da operação anterior (R). Isto significa que podemos dividir um pacote em *N* blocos, calcular o OU exclusivo de todos os blocos e enviar *N* + 1 blocos. Se algum bloco for perdido ou danificado, ele pode ser recriado no lado do receptor. Agora, devemos nos perguntar qual deve ser o valor de *N*. Se *N* = 4, significa que precisamos enviar 25% de dados extras e seremos capazes de corrigir os dados se apenas um dos quatro blocos for perdido.

Entrelaçamento de blocos

Outra maneira de se aplicar FEC em dados multimídia é permitir que alguns pequenos blocos sejam perdidos antes de chegarem ao receptor. Não podemos permitir que todos os blocos pertencentes ao mesmo pacote estejam ausentes; mas, sim, que um bloco de cada pacote esteja ausente. A Figura 8.33 mostra que podemos dividir cada pacote em cinco blocos (normalmente esse número é muito maior). Podemos, então, criar dados bloco a bloco (na horizontal), mas combinar os blocos em pacotes na vertical. Neste caso, cada pacote enviado transporta um bloco proveniente de vários pacotes originais. Se o pacote for perdido, perdemos apenas um bloco de cada pacote, o que normalmente é aceitável no contexto de comunicações multimídia.

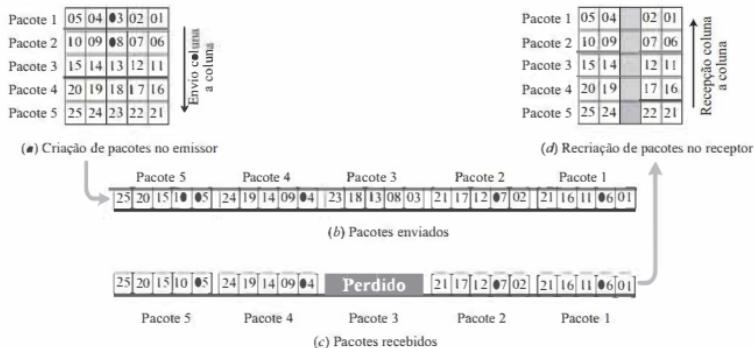



Figura 8.33 Entrelaçamento

Combinando a distância de Hamming e o entrelaçamento

A distância de Hamming e o entrelaçamento podem ser combinados. Podemos começar criando pacotes de *n bits* capazes de corrigir erros em *t bits*. Em seguida, entrelaçamos *m* linhas e enviamos os *bits* coluna a coluna. Dessa forma, podemos corrigir automaticamente erros em rajada que causem erros em até $m \times t$ *bits*.

Compondo pacotes de alta e baixa resolução

Outra solução consiste em criar uma duplicata de cada pacote com uma redundância de baixa resolução e combinar a versão redundante com o próximo pacote. Por exemplo, podemos criar quatro pacotes de baixa resolução a partir de cinco pacotes de alta resolução e enviá-los conforme mostra a Figura 8.34. Se um pacote for perdido, podemos usar a versão de baixa resolução obtida a partir do pacote seguinte. Perceba que a seção de baixa resolução do primeiro pacote fica vazia. Nesse método, se o último pacote for perdido, ele não pode ser recuperado, porém usamos

a versão de baixa resolução de outro pacote se o pacote perdido não for o último. A reprodução de áudio e vídeo não apresentará a mesma qualidade, mas tal falta de qualidade não é percebida na maior parte do tempo.

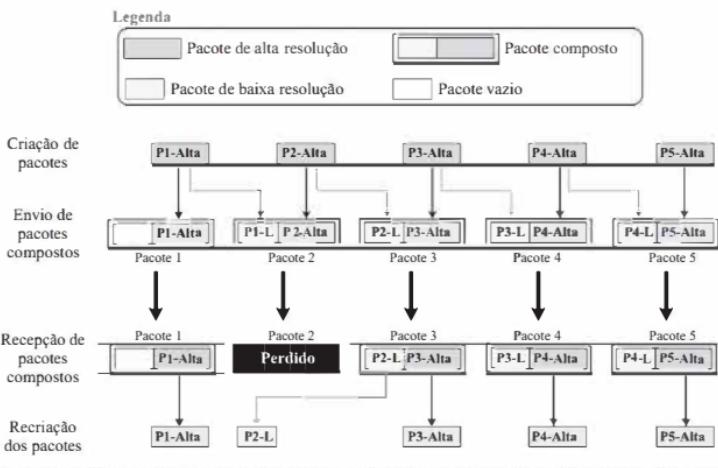



Figura 8.34 Compondo pacotes de alta e baixa resolução.

Exemplo de um aplicativo em tempo real: Skype

O Skype (abreviação do projeto original *Sky peer-to-peer*, ou "Céu par a par") é um *software* de VoIP *peer-to-peer* originalmente desenvolvido por Ahti Heinla, Priit Kasesalu e Jaan Tallinn, os quais também tinham originalmente desenvolvido o Kazaa (um aplicativo para compartilhamento de arquivos via P2P). O aplicativo permite que os usuários registrados, que tenham dispositivos de entrada e saída de áudio em seus PCs, façam chamadas de voz gratuitas, de PC para PC, para outros usuários registrados usando a Internet. O Skype inclui outras ferramentas populares, como a troca de *mensagens instantâneas* (IM – Instant Messaging), Serviço de Mensagens Curtas (SMS – Short Message Service), conversas em grupo, transferência de arquivos, videoconferência e os serviços SkypeIn e SkypeOut. O uso destes permite que os usuários registrados se comuniquem, por uma pequena tarifa, com telefones fixos tradicionais e móveis. Sendo responsável por cerca de 8% dos minutos de chamadas internacionais mundiais no terceiro trimestre de 2009, a empresa Skype é considerada uma das principais corporações globais de comunicação via Internet.

O Skype é gratuito para chamadas de PC para PC*, mas quando uma Rede de Telefonia Pública Comutada (PSTN – Public Switched Telephone Network), que corresponde à rede de telefonia fixa tradicional, ou um telefone celular está envolvido, o Skype oferece um serviço pago. Existem dois modos de empregar um telefone fixo ou celular em uma conversa no Skype: o SkypeIn e o SkypeOut.

* N. de T.: Atualmente, o Skype também está disponível para vários dispositivos móveis, como *smartphones* e *tablets*. As ligações podem ser realizadas/recebidas através de redes sem fio (WiFI, 3G, LTE, etc.) que possuam conexão com a Internet.

O serviço de SkypeIn é oferecido em diversos países do mundo e a lista vem se expandindo. Onde quer que o serviço seja oferecido, ele permite que o usuário cadastrado do Skype receba chamadas provenientes de um telefone fixo ou celular em seu computador, por meio da Internet. Para utilizar o SkypeIn, o usuário se inscreve para receber um número *online* de telefone, pagando uma tarifa mensal. Usando um telefone fixo ou celular, qualquer pessoa pode ligar para esse número e pagar a tarifa-padrão cobrada, como se estivesse fazendo uma chamada semelhante para outro telefone fixo ou celular com o mesmo código de área. O número *online* usa a Internet para rotear a chamada para o usuário registrado do Skype. Exceto pelo preço do número *online*, o custo da chamada é gratuito para o usuário cadastrado que recebe a chamada. Para que a pessoa realizando a chamada não tenha que pagar tarifas de longa distância, o usuário registrado pode se cadastrar em mais de um número *online*. Por exemplo, o usuário registrado pode possuir um número nos Estados Unidos e outro na França. O SkypeIn vem com um serviço gratuito de correio de voz (secretária eletrônica).

O SkypeOut permite que usuários registrados do Skype façam chamadas telefônicas de seus PCs para qualquer telefone fixo ou celular em qualquer lugar do mundo, pagando tarifas locais. Para fazer chamadas com o SkypeOut, o usuário pode adquirir uma assinatura mensal ou então comprar minutos de crédito Skype. Usando um PC, o usuário disca o número de telefone fixo ou celular desejado. O Skype transfere chamadas do SkypeOut para *gateways* que, então, direcionam as chamadas para o serviço de telefonia fixa ou celular. Além da tarifa cobrada pela assinatura mensal ou pelo custo do crédito de minutos Skype, o usuário paga uma pequena tarifa global e local pelo serviço. Com suas assinaturas, os usuários são capazes de redirecionar chamadas recebidas para seus telefones fixos ou celulares. É importante mencionar que o Skype não é considerado um substituto para o serviço de telefonia e não pode ser utilizado em situações de emergência. Por exemplo, não podemos usar o Skype para discar 911 nos Estados Unidos ou 190 no Brasil.

8.4 PROTOCOLOS INTERATIVOS EM TEMPO REAL

Concluída a discussão sobre as três abordagens de uso de multimídia por meio da Internet, nos concentramos agora na última delas, uma das mais interessantes e complicadas: multimídia interativa em tempo real. Esse tipo de aplicação tem atraído muita atenção no domínio da Internet e diversos protocolos da camada de aplicação foram projetados para lidar com seus requisitos. Antes de discutirmos a necessidade e a lógica por trás desse tipo de aplicação, mostramos uma representação esquemática na Figura 8.35.

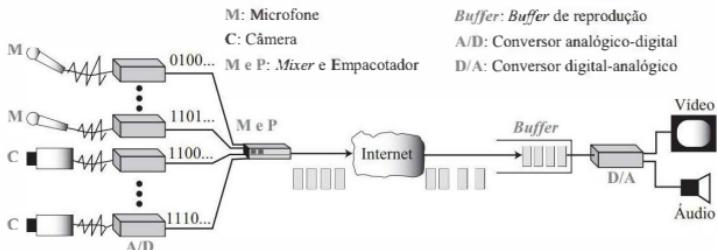



Figura 8.35 Diagrama esquemático de um sistema de multimídia em tempo real.

Embora possa haver um único microfone e um único reprodutor de áudio, as atuais aplicações em tempo real são normalmente compostas por vários microfones e várias câmeras. As

informações de áudio e de vídeo (sinais analógicos) são convertidas em dados digitais, que são criados a partir de diferentes fontes e normalmente mixados (misturados) e empacotados. Os pacotes são enviados para o sistema de comutação de pacotes da Internet. São recebidos no destino com diferentes atrasos (*jitter*) e alguns também podem ser corrompidos ou perdidos. Um *buffer* de reprodução processa os pacotes com base no carimbo de tempo de cada um. O resultado é enviado para um conversor digital-analógico para que os sinais de áudio e de vídeo sejam reproduzidos. O sinal de áudio é enviado para uma caixa de som; o de vídeo é enviado para um dispositivo de exibição (um monitor, por exemplo).

Cada microfone ou câmera no local de origem é denominado um *contribuinte* e recebe um identificador de 32 *bits* denominado *identificador de fonte contribuinte* (CSRC – Contributing Source). O *mixer* é também chamado sincronizador e recebe outro identificador denominado *identificador de fonte de sincronização* (SSRC – Synchronizing Source). Usaremos esses identificadores no pacote mais tarde.

8.4.1 Justificativa para novos protocolos

Discutimos a pilha de protocolos para aplicações em geral da Internet nos Capítulos 2 a 7. Nesta seção, queremos mostrar por que precisamos de alguns novos protocolos para lidar com aplicações multimídia interativas em tempo real, como conferência de áudio e vídeo.

É claro que não precisamos alterar as três primeiras camadas da pilha de protocolos TCP/IP (camadas física, de enlace de dados e de rede), pois elas são projetadas para transportar qualquer tipo de dados. A camada física fornece serviços para a camada de enlace de dados, não importando a natureza dos *bits* em um quadro. A camada de enlace de dados é responsável pela entrega nó a nó dos pacotes provenientes da camada de rede, não importando a composição dos pacotes. A camada de rede também é responsável pela entrega *host* a *host* de um datagrama, não importando o que está nele, embora precisemos de uma camada de rede com uma melhor qualidade de serviço para aplicações multimídia conforme discutiremos na próxima seção.

Parece que devemos nos preocupar apenas com as camadas de aplicação e de transporte. Alguns protocolos da camada de aplicação devem ser projetados para codificar e comprimir os dados multimídia, considerando o equilíbrio entre requisitos de qualidade, largura de banda e complexidade das operações matemáticas de codificação e de compressão. Conforme descreveremos em breve, os protocolos da camada de aplicação capazes de lidar com dados multimídia têm alguns requisitos que podem ser tratados pela camada de transporte e não individualmente por cada protocolo de aplicação.

Camada de aplicação

É claro que precisamos desenvolver alguns protocolos da camada de aplicação para multimídia interativa em tempo real, pois a natureza de aplicações de audioconferência e de videoconferência é diferente daquela de algumas outras aplicações, como transferência de arquivos e correio eletrônico, que discutimos no Capítulo 2. Diversas aplicações proprietárias foram desenvolvidas pelo setor privado e mais e mais aplicações vêm aparecendo no mercado a cada dia. Alguns desses aplicativos, como áudio MPEG e vídeo MPEG, usam alguns padrões definidos para transferência de dados de áudio e de vídeo. Não há um padrão específico usado por todas as aplicações e não há um protocolo de aplicação específico que possa ser utilizado por todos.

Camada de transporte

A falta de um padrão único, juntamente com as características gerais das aplicações multimídia discutidas na Seção 8.3.3, levantam algumas questões sobre o protocolo da camada de transporte a

ser usado por todas as aplicações multimídia. Os dois protocolos comuns da camada de transporte, o TCP e o UDP, foram desenvolvidos em um momento em que ninguém ainda pensava no uso de multimídia na Internet. Podemos usar o UDP ou o TCP como o protocolo da camada de transporte para aplicações de multimídia em tempo real em geral? Para responder a essa pergunta, primeiramente precisamos considerar os requisitos desse tipo de aplicação multimídia e então verificar se o UDP ou o TCP são capazes de satisfazer esses requisitos.

Requisitos da camada de transporte para multimídia interativa em tempo real

Primeiramente, construiremos brevemente um conjunto de requisitos para esse tipo de aplicação.

- **Negociação entre emissor e receptor.** O primeiro requisito refere-se à falta de um padrão único de áudio ou vídeo. Existem vários padrões possíveis para uso em audioconferência e videoconferência, cada um tendo diferentes métodos de codificação e de compressão. Se um emissor utiliza um método de codificação e o receptor usa outro, então a comunicação é impossível. Os aplicativos precisam negociar os padrões utilizados para áudio/vídeo antes que os dados codificados e comprimidos possam ser transferidos.
- **Criação do fluxo de pacotes.** Quando discutimos o UDP e o TCP no Capítulo 3, mencionamos que o UDP permite que o aplicativo empacote sua mensagem com limites bem definidos antes de entregá-la para o UDP. O TCP, por outro lado, pode lidar com fluxos de *bytes* sem exigir que a aplicação imponha limites específicos aos blocos de dados. Em outras palavras, o UDP é adequado para aplicações que precisam enviar mensagens com limites bem definidos, enquanto o TCP é adequado para aplicações que enviam fluxos contínuos de *bytes*. No contexto de multimídia em tempo real, precisamos de ambos os recursos. Multimídia em tempo real consiste em um *fluxo de quadros* ou um *fluxo de blocos* de dados no qual o bloco ou o quadro tem um tamanho ou limites específicos, mas também há uma relação entre os quadros ou blocos. É evidente que o UDP e o TCP não são adequados para lidar com fluxos de quadros nesse caso. O UDP não é capaz de fornecer uma relação entre os quadros; o TCP fornece uma relação entre os *bytes*, mas um *byte* é muito menor do que um quadro ou bloco multimídia.
- **Sincronização de fontes.** Se uma aplicação utiliza mais do que uma fonte de dados (por exemplo, tanto áudio como vídeo), é necessário haver sincronização entre as fontes. Por exemplo, em uma aplicação de teleconferência que utiliza tanto áudio como vídeo, como o Skype, o áudio e o vídeo podem estar usando métodos de codificação diferentes e métodos de compressão com diferentes taxas. É óbvio que, de alguma forma, esses dois tipos de dados devem ser sincronizados; caso contrário, veremos o rosto do locutor antes de ouvir o que ela está dizendo ou vice-versa. Também é possível haver mais do que uma fonte de áudio ou de vídeo (por exemplo, quando são utilizados vários microfones ou múltiplas câmeras). A sincronização de fontes é normalmente feita por meio de *mixers*.
- **Controle de erros.** Já discutimos que tratar erros (corrupção e perda de pacotes) requer cuidados especiais em aplicações multimídia em tempo real. Mostramos que não podemos nos dar ao luxo de retransmitir pacotes corrompidos ou perdidos. Aprendemos que precisamos introduzir redundância extra nos dados para sermos capazes de reproduzir os pacotes perdidos ou danificados sem precisar pedir que eles sejam retransmitidos. Isto implica que o protocolo TCP, que apresenta as características listadas como negativas, não é adequado para aplicações multimídia em tempo real.
- **Controle de congestionamento.** Como em outras aplicações, é preciso fornecer algum tipo de controle de congestionamento em aplicações multimídia. Se decidirmos não usar o TCP para multimídia (em razão dos problemas de retransmissão), devemos, de alguma forma, implementar o controle de congestionamento no sistema.

- **Remoção de *jitter*.** Discutimos na Seção 8.3.3 que um dos problemas no contexto de aplicações multimídia em tempo real é o *jitter* observado no lado do receptor, pois o serviço de comutação de pacotes fornecido pela Internet pode criar atrasos desiguais para os diferentes pacotes em um fluxo. No passado, serviços de audioconferência eram fornecidos pela rede telefônica, que foi originalmente projetada como uma rede de comutação de circuitos, na qual não há *jitter*. Se migrarmos gradualmente todos esses aplicativos para a Internet, precisamos lidar de alguma forma com o *jitter*. Dissemos na Seção 8.3.3 que uma das formas de reduzir o *jitter* é usar *buffers* de reprodução e criar carimbos de tempo. O processo de reprodução é implementado na camada de aplicação no lado do receptor, mas a camada de transporte deve ser capaz de fornecer à camada de aplicação as funcionalidades de carimbo de tempo e sequenciamento.
- **Identificação da origem.** Uma questão sutil em aplicações multimídia, assim como em outras aplicações, é identificar a origem dos dados na camada de aplicação. Quando usamos a Internet, as partes são identificadas por seus endereços IP. No entanto, é preciso mapear os endereços IP para algo mais amigável, como fizemos com o protocolo HTTP ou com o correio eletrônico.

Capacidade do UDP e do TCP em tratar multimídia em tempo real

Agora que discutimos os requisitos de multimídia em tempo real, veremos se UDP e o TCP são capazes de satisfazer esses requisitos. A Tabela 8.6 compara o UDP e o TCP com relação a esses requisitos.

À primeira vista, a Tabela 8.6 revela um fato muito interessante: nem o UDP nem o TCP são capazes de satisfazer todos os requisitos. No entanto, devemos lembrar que precisamos de um protocolo da camada de transporte para implementar o *socket* cliente-servidor; não podemos deixar a camada de aplicação fazer o trabalho da camada de transporte. Isto significa que, provavelmente, temos três opções:

1. Podemos usar um novo protocolo de camada de transporte (como o SCTP, discutido no final deste capítulo), que combina as características do UDP e do TCP (em particular, o empacotamento de fluxos e suporte a múltiplos fluxos). Esta é provavelmente a melhor escolha, porque o SCTP combina as características do UDP e do TCP com seus próprios recursos adicionais. Entretanto, o SCTP foi criado quando já havia muitas aplicações multimídia. Ele pode se tornar a camada de transporte *de facto* no futuro.

Tabela 8.6 Capacidade do UDP e do TCP em lidar com multimídia em tempo real.

Requisitos	UDP	TCP
1. Negociação entre emissor e receptor para selecionar o tipo de codificação	Não	Não
2. Criação de fluxo de pacotes	Não	Não
3. Sincronização de fontes para mixar diferentes fontes	Não	Não
4. Controle de erros	Não	Sim
5. Controle de congestionamento	Não	Sim
6. Remoção de *jitter*	Não	Não
7. Identificação da origem	Não	Não

2. Podemos usar o TCP e combiná-lo com outros mecanismos de transporte para compensar pelos requisitos que não podem ser satisfeitos pelo TCP. No entanto, essa escolha é um pouco difícil porque o TCP usa um método de retransmissão que não é aceitável para aplicações em tempo real. Outro problema com o TCP é que ele não faz *multicast*. Uma conexão TCP envolve apenas duas partes; precisamos de uma conexão com múltiplas partes para comunicação interativa em tempo real.
3. Podemos usar o UDP e combiná-lo com outros mecanismos de transporte para compensar os requisitos que não podem ser satisfeitos pela UDP. Em outras palavras, usamos o UDP para fornecer uma interface de *socket* entre o cliente e o servidor, porém usamos outro protocolo que é executado sobre o UDP. Essa é a opção atual usada por aplicações multimídia. Esse mecanismo de transporte é o Protocolo de Transporte em Tempo Real (RTP – Real-time Transport Protocol), que discutimos a seguir.

8.4.2 RTP

O **Protocolo de Transporte em Tempo Real** (RTP – Real-time Transport Protocol) é o protocolo concebido para lidar com tráfego em tempo real na Internet. O RTP não tem um mecanismo de entrega (*multicast*, números de porta e assim por diante); ele deve ser usado com o UDP. O RTP fica localizado entre o UDP e a aplicação multimídia. A literatura e os padrões existentes tratam o RTP como um protocolo de transporte (não um protocolo da camada de transporte), que pode ser visto como estando localizado na camada de aplicação (ver Figura 8.36). Os dados provenientes de aplicações multimídia são encapsulados no RTP, que, por sua vez, os passa para a camada de transporte. Em outras palavras, a interface *socket* está localizada entre o RTP e o UDP, o que implica que devemos incluir a funcionalidade do RTP em programas cliente-servidor que escrevemos para cada aplicação multimídia. No entanto, algumas linguagens de programação oferecem certos recursos para facilitar essa tarefa de programação. Por exemplo, a linguagem C fornece uma biblioteca RTP e a linguagem Java fornece uma classe RTP para isso. Se usarmos a biblioteca RTP ou a classe RTP, podemos pensar como se as aplicações fossem separadas do RTP em si, como se o RTP fosse parte da camada de transporte.

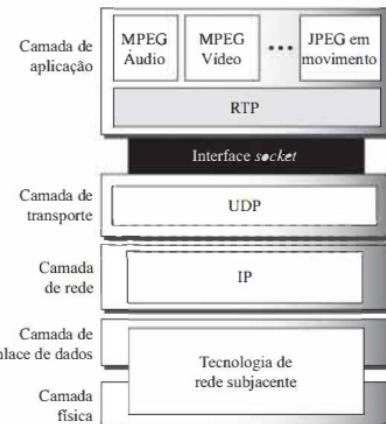



Figura 8.36 Localização do RTP na pilha de protocolos TCP/IP.

Formato dos pacotes RTP

Antes de discutirmos como o RTP pode ajudar as aplicações multimídia, abordaremos o formato de seus pacotes. Podemos, então, relacionar as funções dos campos com os requisitos que discutimos na seção anterior. A Figura 8.37 mostra o formato do cabeçalho dos pacotes RTP. O formato é muito simples e genérico o suficiente para cobrir todas as aplicações em tempo real. Uma aplicação que requer mais informações pode inseri-las no início da carga útil do pacote.

Ver	P	X	Contagem de contr.	M	Tipo de carga útil	Número de sequência
Carimbo de tempo						
Identificador de fonte de sincronização (SSRC)						
Identificador de fonte contribuinte (CSRC) (1)						
⋮						
Identificador de fonte contribuinte (CSRC) (N)						
Cabeçalho de extensão						

Figura 8.37 Formato de cabeçalho dos pacotes RTP.

Uma descrição de cada campo é dada a seguir

- **Ver.** Campo de 2 *bits* que especifica o número da versão. A versão atual é a 2.
- **P.** Campo de 1 *bit* que, se valer 1, indica a presença de *bytes* de enchimento (*padding*) no final do pacote. Nesse caso, o valor do último *byte* no enchimento indica o comprimento deste. A inserção de *bytes* de enchimento é obrigatória caso o pacote seja cifrado. Não há qualquer enchimento se o valor do campo P for 0. O uso desse campo de 1 *bit* elimina a necessidade de explicitar o comprimento dos dados RTP porque, se não houver *bytes* de enchimento, o comprimento dos dados é igual ao comprimento dos dados UDP menos o tamanho do cabeçalho RTP. Caso contrário, o comprimento dos *bytes* de enchimento deve ser subtraído para se obter o comprimento dos dados RTP.
- **X.** Campo de 1 *bit*. Se valer 1, indica a existência de um cabeçalho de extensão entre o cabeçalho básico e os dados. Não há qualquer cabeçalho de extensão se o valor desse campo for 0.
- **Contador de contribuintes.** Campo de 4 *bits* que define o número de fontes contribuintes (CSRC). Perceba que podemos ter um máximo de 15 contribuintes, pois um campo de 4 *bits* só comporta um número entre 0 e 15. Note que em uma conferência de áudio ou de vídeo, cada fonte ativa (a fonte que envia dados em vez de apenas ouvir) é chamada de contribuinte.
- **M.** Campo de 1 *bit* que é um marcador utilizado pela aplicação para indicar, por exemplo, o fim de seus dados. Dissemos que uma aplicação multimídia envia um fluxo de blocos ou quadros com um marcador de fim de quadro. Se esse *bit* valer 1 em um pacote RTP, significa que o pacote RTP carrega tal marcador.
- **Tipo de carga útil.** Campo de 7 *bits* que indica o tipo de carga útil. Vários tipos de carga útil foram definidos até hoje. Listamos algumas aplicações comuns na Tabela 8.7. Uma discussão sobre os tipos possíveis está além do escopo deste livro.

Tabela 8.7 Tipos de cargas úteis.

Tipo	Aplicação	Tipo	Aplicação	Tipo	Aplicação
0	Áudio PCMμ	7	Áudio LPC	15	Áudio G728
1	1016	8	Áudio PCMA	26	JPEG em movimento
2	Áudio G721	9	Áudio G722	31	H.261
3	Áudio GSM	10–11	Áudio L16	32	Vídeo MPEG1
5–6	Áudio DV14	14	Áudio MPEG	33	Vídeo MPEG2

- **Número de sequência.** Campo que tem 16 *bits* de comprimento. Ele é usado para enumerar os pacotes RTP. O número de sequência do primeiro pacote é escolhido aleatoriamente; ele é incrementado de 1 para cada pacote subsequente. O número de sequência é usado pelo receptor para detectar pacotes perdidos ou fora de ordem.
- **Carimbo de tempo (*timestamp*).** Campo de 32 *bits* que indica a relação temporal entre pacotes. O carimbo de tempo do primeiro pacote é um número aleatório. Para cada pacote subsequente, o valor usado corresponde à soma do carimbo de tempo anterior com o tempo decorrido desde que o primeiro *byte* foi produzido (amostrado). O número de pulsos do relógio depende da aplicação. Por exemplo, aplicativos de áudio normalmente geram blocos de 160 *bytes*; o número de pulsos do relógio nessa aplicação é 160. O carimbo de tempo para essa aplicação é incrementado de 160 para cada pacote RTP.
- **Identificador de fonte de sincronização (SSRC).** Se houver apenas uma fonte, esse campo de 32 *bits* a define. Entretanto, se houver várias fontes, o *mixer* é a fonte de sincronização e as outras fontes são contribuintes. O valor do identificador de uma fonte é um número aleatório escolhido por ela. O protocolo fornece uma estratégia em caso de conflito (quando duas fontes tentam usar um mesmo número de sequência).
- **Identificador de fonte contribuinte (CSRC).** Cada um desses identificadores de 32 *bits* (no máximo 15) define uma fonte. Quando há mais do que uma fonte em uma sessão, o *mixer* é a fonte de sincronização e as fontes restantes são contribuintes.

Porta UDP

Como o RTP em si não é um protocolo da camada de transporte, o pacote RTP não é encapsulado diretamente em um datagrama IP. Em vez disso, o RTP é tratado como um programa da camada de aplicação e é encapsulado em um datagrama de usuário UDP. No entanto, contrariamente a outros programas da camada de aplicação, nenhuma porta bem conhecida é atribuída ao RTP. A porta pode ser selecionada sob demanda com uma única restrição: o número dela deve ser par. O número subsequente (ímpar) é usado pelo protocolo parceiro do RTP, o Protocolo de Controle de Transporte em Tempo Real (RTCP – Real-time Transport Control Protocol), que discutiremos na próxima seção.

O RTP usa um número de porta UDP par.

8.4.3 RTCP

O RTP suporta apenas um tipo de mensagem, que carrega os dados da origem para o destino. Para realmente controlar a sessão, precisamos de uma maior comunicação entre os participantes em uma sessão. A comunicação de controle nesse caso é atribuída a um protocolo separado, denominado

Protocolo de Controle de Transporte em Tempo Real (RTCP – Real-time Transport Control Protocol). Precisamos enfatizar que a carga útil do RTCP não é transportada em um pacote RTP; o RTCP é, na verdade, um protocolo "irmão" do RTP. Isto significa que o UDP, como o protocolo de transporte de fato, às vezes carrega cargas úteis do RTP e às vezes carrega cargas úteis do RTCP, como se elas pertencessem a diferentes protocolos da camada superior.

Os pacotes RTCP criam um fluxo de controle ***fora da banda*** que fornece informações de retorno entre os emissores e receptores das transmissões multimídia, em ambas as direções. Em particular, o RTCP fornece as seguintes funcionalidades:

1. Informa o emissor ou emissores de fluxos multimídia a respeito do desempenho da rede, algo que pode estar diretamente relacionado ao congestionamento desta. Como as aplicações multimídia usam UDP (em vez de TCP), não há uma forma de controlar o congestionamento na rede na camada de transporte. Ou seja, se for necessário controlar o congestionamento, isto deve ser feito na camada de aplicação. O RTCP, conforme veremos em breve, fornece as pistas para que a camada de aplicação faça isso. Se o congestionamento for observado e reportado pelo RTCP, uma aplicação pode utilizar um método de compressão mais agressivo para reduzir o número de pacotes e, portanto, reduzir o congestionamento em troca de uma redução de qualidade. Por outro lado, se nenhum congestionamento for observado, o aplicativo pode usar um método de compressão menos agressivo para obter um serviço de melhor qualidade.
2. A informação transportada nos pacotes RTCP pode ser usada para sincronizar diferentes fluxos associados à mesma fonte. Uma fonte pode usar duas fontes diferentes para coletar dados de áudio ou vídeo. Além disso, dados de áudio podem ser coletados a partir de diferentes microfones e dados de vídeo podem ser coletados a partir de diferentes câmeras. Em geral, duas informações são necessárias para conseguir a sincronização:
   a. Cada emissor precisa de uma identidade. Embora cada fonte possa ter um SSRC diferente, o RTCP fornece uma única identidade, conhecida como *Nome Canônico* (CNAME – Canonical Name), para cada fonte. O CNAME pode ser usado para correlacionar diferentes fontes e permitir que o receptor combine fontes diferentes de um mesmo emissor. Por exemplo, uma teleconferência pode ter *n* emissores associados a uma sessão, porém *m* fontes ($m > n$) que contribuem para o fluxo. Nesse sistema, temos apenas *n* CNAMEs, mas *m* valores de SSRC. Um CNAME é escrito na forma

      usuário@host

      onde *usuário* é normalmente o nome de registro do usuário e *host* é o nome de domínio do *host*.
   b. O nome canônico não pode, por si só, fornecer sincronização. Para sincronizar as fontes, precisamos conhecer a temporização absoluta do fluxo, além da temporização relativa fornecida pelo campo de carimbo de tempo em cada pacote RTP. A informação do carimbo de tempo em cada pacote fornece a relação temporal relativa dos *bits* no pacote com relação ao início do fluxo; ela não é capaz de relacionar um fluxo a outro. O tempo absoluto, ou tempo de "relógio de parede", como também é conhecido, deve ser enviado por pacotes RTCP para permitir a sincronização.
3. Um pacote RTCP pode transportar informações extras sobre o emissor que podem ser úteis para o receptor, como o nome do emissor (além do nome canônico) ou legendas para um vídeo.

Pacotes RTCP

Agora que discutimos as principais funcionalidades e finalidades do RTCP, discutiremos seus pacotes. A Figura 8.38 mostra cinco tipos de pacotes comuns. O número ao lado de cada caixa

indica o valor numérico de cada pacote. Precisamos mencionar que alguns pacotes RTCP podem ser empacotados como uma única carga útil UDP, pois os pacotes RTCP são menores do que os pacotes RTP.

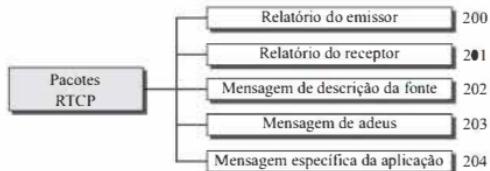



Figura 8.38 Tipos de pacotes RTCP.

O formato e a definição exata de cada campo são bastante complexos e vão além do escopo e do espaço deste livro. Discutimos brevemente a finalidade de cada pacote e a relacionamos com as funcionalidades descritas anteriormente.

Pacote de relatório do emissor

O pacote de relatório do emissor é enviado periodicamente pelos emissores ativos em uma sessão para relatar estatísticas de transmissão e recepção de todos os pacotes RTP enviados durante o intervalo. O pacote de relatório do emissor inclui as seguintes informações:

- O SSRC do fluxo RTP.
- O carimbo de tempo absoluto, que é a combinação do carimbo de tempo relativo com o tempo real (tempo de relógio de parede), que é o número de segundos decorridos desde a meia-noite de 1º de janeiro de 1970. O carimbo de tempo absoluto, conforme discutido anteriormente, permite que o receptor sincronize diferentes pacotes RTP.
- O número de pacotes RTP e *bytes* enviados desde o início da sessão.

Pacote de relatório do receptor

O relatório do receptor é enviado por participantes passivos, aqueles que não enviam pacotes RTP. O relatório informa o emissor e outros receptores sobre a qualidade do serviço. Essa informação de retorno pode ser usada para controlar o congestionamento no lado do emissor. Um relatório do receptor inclui as seguintes informações:

- O SSRC do fluxo RTP para o qual o relatório do receptor foi gerado.
- A porcentagem de perda de pacotes.
- O último número de sequência.
- Uma estimativa estatística do *jitter* entre recepções.

Pacote de descrição da fonte

A fonte envia periodicamente um pacote de descrição da fonte para fornecer informações adicionais sobre si mesma. Ele pode incluir:

- O SSRC.
- O nome canônico (CNAME) do emissor.

- Outras informações, como o nome real, o endereço de e-mail e o número de telefone.
- Dados adicionais, como legendas para o vídeo.

Pacote de adeus

Uma fonte envia um pacote de gerenciamento de membros para encerrar um fluxo. Ele permite que a fonte anuncie que ela está deixando a conferência. Embora outras fontes possam detectar a ausência de uma fonte, esse pacote é um anúncio direto. Ele também é muito útil para um *mixer*.

Pacote específico da aplicação

O pacote específico da aplicação é um pacote empregado por um aplicativo que quer usar novas aplicações (não definidas no padrão). Ele permite a definição de um novo tipo de pacote.

Porta UDP

O RTCP, assim como o RTP, não usa uma porta UDP bem conhecida, mas, sim, uma porta temporária. O número de porta UDP escolhido deve ser aquele imediatamente posterior ao número de porta UDP selecionado para o RTP, o que faz dele um número de porta ímpar.

O RTCP usa um número de porta UDP ímpar que sucede o número escolhido para o RTP.

Utilização de banda

Os pacotes RTCP são enviados não apenas pelos emissores ativos, mas também por receptores passivos, cujos números são normalmente maiores do que aqueles dos emissores ativos. Isto significa que se o tráfego RTCP não for controlado, ele pode tornar-se elevado. Para controlar a situação, o RTCP utiliza um mecanismo de controle para limitar o seu tráfego a uma pequena parcela (normalmente 5%) do tráfego utilizado na sessão (tanto pelo RTP como pelo RTCP). A maior parte dessa pequena porcentagem, de x%, é, então, atribuída aos pacotes RTCP gerados pelos receptores passivos, enquanto uma parte menor, de $(1 - x)$%, é atribuída aos pacotes RTCP gerados pelos emissores ativos. O protocolo RTCP usa um mecanismo para definir o valor de x com base na proporção entre receptores passivos e emissores ativos.

Exemplo 8.10

Consideremos que a largura de banda total alocada para uma sessão seja de 1 Mbps. O tráfego RTCP recebe apenas 5% dessa banda, o que dá 50 Kbps. Se houver apenas dois emissores ativos e oito receptores passivos, é natural que cada emissor ou receptor fique com apenas 5 Kbps. Se o tamanho médio do pacote RTCP for de 5 Kbits, então, cada emissor ou receptor pode enviar apenas um pacote RTCP por segundo. Perceba que é preciso considerar o tamanho do pacote na camada de enlace de dados.

Satisfação dos requisitos

Conforme prometemos, vejamos como a combinação do RTP e do RTCP pode atender as exigências de uma aplicação multimídia interativa em tempo real. Um fluxo contínuo de áudio ou vídeo digital, que corresponde a uma sequência de *bits*, é dividido em blocos (*pedaços* ou *quadros*, como são geralmente chamados). Cada bloco tem limites predefinidos que distinguem

um bloco do bloco anterior ou do próximo. Um bloco é encapsulado em um pacote RTP, que define uma codificação específica (tipo de carga útil), número de sequência, carimbo de tempo, identificador de fonte de sincronização (SSRC) e um ou mais identificadores de fontes contribuintes (CSRC).

1. O primeiro requisito, a *negociação entre emissor e receptor*, não pode ser satisfeito pela combinação dos protocolos RTP e RTCP. Ele deve ser atendido por outros meios. Veremos mais adiante que outro protocolo (o SIP), que é usado em conjunto com o RTP e o RTCP, fornece esse recurso.
2. O segundo requisito, a *criação de um fluxo de blocos*, é fornecido encapsulando-se cada bloco em um pacote RTP e atribuindo um número de sequência para eles. O campo M em um pacote RTP também define se existe um tipo específico de delimitação entre os blocos.
3. O terceiro requisito, a *sincronização de fontes*, é satisfeito por meio da identificação de cada fonte por um identificador de 32 *bits* e usando-se o carimbo de tempo relativo no pacote RTP e o carimbo de tempo absoluto no pacote RTCP.
4. O quarto requisito, o *controle de erros*, é fornecido usando o número de sequência no pacote RTP e permitindo-se que o aplicativo regenere o pacote perdido usando métodos de FEC discutidos anteriormente neste capítulo.
5. O quinto requisito, o *controle de congestionamento*, é satisfeito por meio das mensagens de retorno do receptor usando os *pacotes de relatório do emissor* (RTCP) que notificam o emissor sobre o número de pacotes perdidos. O emissor pode, então, utilizar uma técnica de compressão mais agressiva para reduzir o número de pacotes enviados e, assim, aliviar o congestionamento.
6. O sexto requisito, a *remoção de jitter*, é obtido por meio do carimbo de tempo e do sequenciamento fornecidos em cada pacote RTP, que são utilizados na reprodução dos dados a partir do *buffer*.
7. O sétimo requisito, a *identificação da origem*, é fornecido por meio do CNAME incluído nos pacotes de descrição da fonte (RTCP) enviados pelo emissor.

8.4.4 Protocolo de Iniciação de Sessão

Discutimos como usar a Internet para fazer conferências de áudio e vídeo. Embora o RTP e o RTCP possam ser usados para fornecer esses serviços, um componente está faltando: um sistema de sinalização necessário para chamar os participantes. Para entender o problema, voltemos por ora à conferência de áudio tradicional (entre duas ou mais pessoas), utilizando o sistema de telefonia tradicional (a rede pública de telefonia fixa). Para fazer uma chamada telefônica, dois números de telefone são necessários: o da pessoa que faz a chamada (emissor) e o da pessoa que a recebe (receptor). Precisamos, então, discar o número de telefone do receptor e esperar que ele responda. A conversação telefônica se inicia após a resposta do receptor. Em outras palavras, a comunicação telefônica regular envolve duas fases: a fase de sinalização e a de comunicação por áudio.

A fase de sinalização na rede de telefonia é fornecida por um protocolo denominado *Sistema de Sinalização 7* (SS7). O protocolo SS7 é totalmente separado do sistema de comunicação de voz. Por exemplo, embora o sistema de telefonia tradicional use sinais analógicos para o transporte de voz sobre uma rede de comutação de circuitos, o SS7 utiliza pulsos elétricos de modo que cada número discado é transformado em uma série de pulsos. Atualmente, o SS7 não fornece somente o serviço de chamada, mas também outros serviços, como a transferência de chamadas e relatório de erros.

A combinação dos protocolos RTP e RTCP que discutimos nas seções anteriores é equivalente à comunicação de voz fornecida pela rede de telefonia fixa tradicional; para simular completamente esse sistema usando a Internet, precisamos de um sistema de sinalização. Nossa ambição nos leva ainda mais longe. Não queremos apenas ser capazes de chamar outra parte para uma conferência de áudio ou vídeo usando nossos computadores (PCs), mas também fazê-lo usando nosso aparelho telefônico, telefone celular, PDAs e assim por diante. Também precisamos encontrar a pessoa chamada se ela não estiver sentada a sua mesa. Precisamos de comunicação entre uma diversidade de dispositivos.

O **Protocolo de Iniciação de Sessão** (SIP – Session Initiation Protocol) é um protocolo desenvolvido pela IETF para ser usado em conjunto com o RTP e o SCTP. Trata-se de um protocolo da camada de aplicação, semelhante ao HTTP, que estabelece, gerencia e encerra uma sessão multimídia (chamada). Pode ser usado para criar sessões com dois participantes, múltiplos participantes ou sessões *multicast*. O SIP foi projetado para ser independente da camada de transporte subjacente; pode ser executado sobre UDP, TCP ou SCTP, usando a porta 5060. O SIP é capaz de fornecer os seguintes serviços:

- Estabelecer uma chamada entre usuários se eles estiverem conectados à Internet.
- Determinar a localização dos usuários (seus endereços IP) na Internet, pois eles podem estar mudando de endereço IP (considere a situação de IP móvel ou o DHCP).
- Descobre se os usuários são capazes ou estão dispostos a participar da conferência.
- Determina a capacidade dos usuários em termos de mídia a ser utilizada e tipo de codificação (o primeiro requisito para comunicação multimídia que mencionamos na seção anterior).
- Permite a configuração da sessão, definindo parâmetros, como números de porta a serem utilizados (lembre-se de que o RTP e o RTCP usam números de porta efêmeros).
- Fornece funções de gerenciamento de sessão, como colocar chamadas em espera, redirecionar chamadas, aceitar novos participantes e alterar os parâmetros da sessão.

Partes comunicantes

Uma diferença que podemos notar entre as aplicações de multimídia interativa em tempo real e outras aplicações refere-se às partes se comunicando. Em uma conferência de áudio ou vídeo, a comunicação se dá entre seres humanos, não entre dispositivos. Por exemplo, no HTTP ou no FTP, o cliente precisa determinar o endereço IP do servidor (usando DNS) antes da comunicação. Não é preciso localizar uma pessoa antes do início da comunicação. No SMTP, o remetente de um e-mail envia a mensagem para a caixa de correio de um destinatário em um servidor SMTP sem controlar quando ela será acessada. Em uma conferência de áudio ou vídeo, a pessoa que faz a chamada precisa encontrar a pessoa chamada. A pessoa chamada pode estar sentada em sua mesa, caminhando na rua ou totalmente indisponível. O que torna a comunicação mais difícil é que o dispositivo ao qual o participante tem acesso em um instante específico pode ter recursos diferentes daqueles disponíveis em outro dispositivo usado em outro instante. O protocolo SIP precisa determinar a localização da pessoa chamada e, ao mesmo tempo, negociar os recursos dos dispositivos que os participantes estão usando.

Endereços

Em uma comunicação telefônica comum, um número de telefone identifica o emissor e outro número de telefone identifica o receptor da chamada. O SIP é muito flexível. No SIP, um endereço de e-mail, um endereço IP, um número de telefone e outros tipos de endereços podem ser usados para identificar o emissor e o receptor. No entanto, o endereço precisa estar no formato SIP (também chamado *esquema*). A Figura 8.39 mostra alguns formatos comuns.

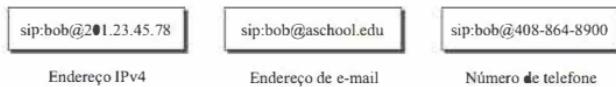



Figura 8.39 Formatos SIP

Percebe-se que o endereço SIP é similar a um URL, que discutimos no Capítulo 2. Na realidade, os endereços SIP são URLs que podem ser incluídos na página Web do receptor em potencial. Por exemplo, Bob pode incluir um dos endereços anteriores como seu endereço SIP e, se alguém clicar sobre ele, o protocolo SIP é invocado e faz uma chamada para Bob. Outros endereços também são possíveis, como endereços usando o primeiro nome da pessoa seguido pelo seu sobrenome, mas todos os endereços precisam estar no formato *sip: usuário@endereço*.

Mensagens

O SIP é um protocolo baseado em texto, como o HTTP. O SIP, assim como o HTTP, usa mensagens. As mensagens no SIP são divididas em duas grandes categorias: pedidos e respostas. O formato de ambas as categorias de mensagens é apresentado a seguir (note a semelhança com as mensagens HTTP, conforme mostrado na Figura 2.12):

Mensagens de pedido		Mensagens de resposta	
Linha inicial		**Linha de estado**	
Cabeçalho	// uma ou mais linhas	**Cabeçalho**	// uma ou mais linhas
Linha em branco		**Linha em branco**	
Corpo	// uma ou mais linhas	**Corpo**	// uma ou mais linhas

Mensagens de pedido

A IETF definiu originalmente seis mensagens de pedido, mas algumas novas foram propostas para estender as funcionalidades do SIP. Mencionamos apenas as seis mensagens originais a seguir:

- **INVITE.** A mensagem INVITE (convite) é uma mensagem de pedido usada pela pessoa que faz a chamada, para inicializar uma sessão. Usando essa mensagem de pedido, quem chama convida uma ou mais pessoas (receptores) a participar da conferência.
- **ACK.** A mensagem ACK (de *acknowledgement*, ou confirmação) é enviada pela pessoa que faz a chamada para confirmar que a inicialização da sessão foi concluída.
- **OPTIONS.** A mensagem OPTIONS (opções) consulta uma máquina sobre suas capacidades.
- **CANCEL.** A mensagem CANCEL (cancelar) cancela um processo de inicialização que já começou, mas não finaliza a chamada. Uma nova inicialização pode ser feita após a mensagem CANCEL.
- **REGISTER.** A mensagem REGISTER (registrar) cria uma conexão quando o receptor da chamada não está disponível.
- **BYE.** A mensagem BYE (adeus) é usada para finalizar a sessão. Podemos comparar a mensagem BYE com a mensagem CANCEL. A mensagem BYE, que pode ser enviada pela parte que chama ou pela parte chamada, finaliza a sessão inteira.

Mensagens de resposta

A IETF também definiu seis tipos de mensagens de resposta que podem ser enviados para solicitar mensagens, mas perceba que não há uma relação entre uma mensagem de pedido e uma de resposta. Uma mensagem de resposta pode ser enviada para qualquer mensagem de pedido. Assim como outros protocolos da camada de aplicação orientados a texto, as mensagens de resposta são definidas usando números de três dígitos. As mensagens de resposta são descritas brevemente a seguir:

- **Respostas informativas.** Essas respostas são da forma **SIP 1xx**. As mais comuns são 100 *trying* (tentando), 180 *ringing* (tocando), 181 *call forwarded* (chamada redirecionada), 182 *queued* (colocada na fila) e 183 *session progress* (progresso da sessão).
- **Respostas bem-sucedidas.** Essas respostas são da forma **SIP 2xx**. A mais comum é 200 OK.
- **Respostas de redirecionamento.** Essas respostas são da forma **SIP 3xx**. As mais comuns são 301 *moved permanently* (movido permanentemente), 302 *moved temporarily* (movido temporariamente) e 380 *alternative service* (serviço alternativo).
- **Respostas de falha do cliente.** Essas respostas são da forma **SIP 4xx**. As mais comuns são 400 *bad request* (pedido inválido), 401 *unauthorized* (não autorizado), 403 *forbidden* (proibido), 404 *not found* (não encontrado), 405 *method not allowed* (método não permitido), 406 *not acceptable* (não aceitável), 415 *unsupported media type* (tipo de mídia não suportado), 420 *bad extension* (extensão inválida) e 486 *busy here* (ocupado aqui).
- **Respostas de falha do servidor.** Essas respostas são da forma **SIP 5xx**. As mais comuns são 500 *internal server error* (erro interno no servidor), 501 *not implemented* (não implementado), 503 *service unavailable* (serviço indisponível), 504 *timeout* (tempo-limite expirado) 505 *SIP version not supported* (versão do SIP não suportada).
- **Respostas de falha globais.** Essas respostas são da forma **SIP 6xx**. As mais comuns são de 600 *busy everywhere* (ocupado em todos os lugares), 603 *decline* (recusada), 604 *doesn't exist* (não existe) e 606 *not acceptable* (não aceitável).

Primeiro cenário: Sessão simples

No primeiro cenário, consideramos que Alice precisa chamar Bob e que a comunicação usa os endereços IP de Alice e de Bob como os endereços SIP. Podemos dividir a comunicação em três módulos: o estabelecimento, a comunicação em si e seu encerramento. A Figura 8.40 mostra uma sessão simples usando SIP.

Estabelecendo uma sessão

Estabelecer uma sessão no SIP exige uma apresentação em três vias, ou *three-way handshake*. Alice envia uma mensagem de pedido do tipo INVITE, usando UDP, TCP ou SCTP, para iniciar a comunicação. Se Bob estiver disposto a iniciar a sessão, ele envia uma mensagem de resposta "200 OK". Para confirmar que um código de resposta foi recebido, Alice envia uma mensagem de pedido do tipo ACK para iniciar a comunicação de áudio. O processo de estabelecimento da sessão usa duas mensagens de pedido (INVITE e ACK) e uma mensagem de resposta (200 OK). Discutiremos sobre o conteúdo das mensagens mais adiante, mas, no momento, precisamos dizer que a primeira linha da mensagem de INVITE contém o endereço IP do receptor e a versão do SIP. Não mostramos qualquer linha no cabeçalho, mas mostraremos mais tarde. O corpo do cabeçalho usa outro protocolo,o Protocolo de Descrição da Sessão (SDP – Session Description Protocol), que define a sintaxe (formato) e a semântica (significado) de cada linha. Discutiremos brevemente esse protocolo mais adiante. Por ora, mencionamos apenas que a primeira linha do

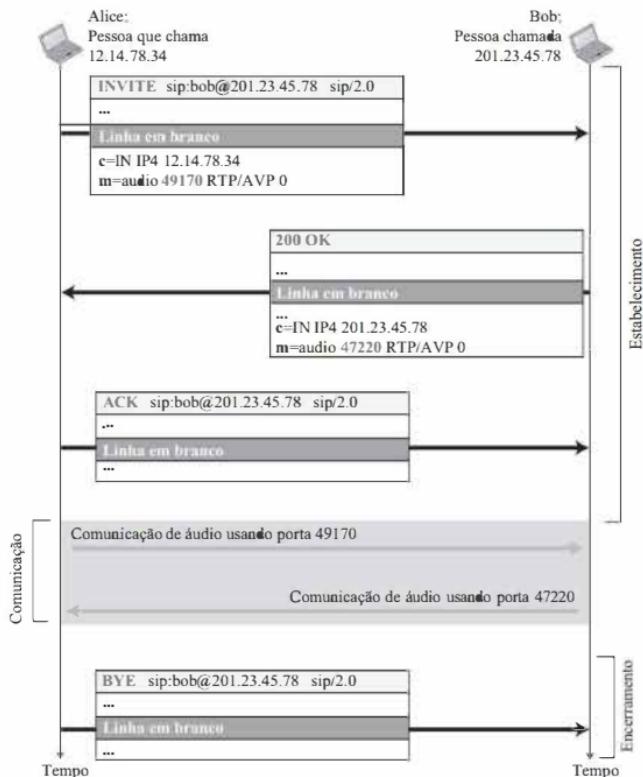



Figura 8.40 Sessão de SIP simples.

corpo da mensagem define seu remetente; a segunda linha define a mídia (áudio) e o número de porta a ser utilizada pelo RTP na direção de Alice para Bob. A mensagem de resposta define a mídia (áudio) e o número de porta a ser utilizado pelo RTP na direção de Bob para Alice. Após Alice confirmar o estabelecimento da sessão com uma mensagem de pedido do tipo ACK (que não requer uma resposta), o estabelecimento da sessão termina e a comunicação pode começar.

Comunicação

Após a sessão ter sido estabelecida, Alice e Bob podem se comunicar usando duas portas temporárias definidas durante o estabelecimento da sessão. As portas pares são usadas pelo RTP; o RTCP pode usar as portas ímpares que vêm em seguida (mostramos apenas as portas pares utilizadas pelo RTP na Figura 8.40).

Encerrando a sessão

A sessão pode ser encerrada com uma mensagem de BYE enviada por qualquer uma das partes. Na figura, consideramos que Alice encerra a sessão.

Segundo cenário: Localizando a pessoa chamada

O que acontece se Bob não estiver próximo ao seu terminal? Ele pode estar longe de seu sistema ou em outro terminal. Pode nem mesmo ter um endereço de IP fixo se o DHCP estiver sendo usado. O SIP tem um mecanismo (similar àquele do DNS) que localiza o endereço IP do terminal no qual Bob pode ser encontrado. Para fazer esse rastreamento, o SIP usa o conceito de registro e define alguns servidores como servidores de registro. A qualquer momento, o usuário está registrado junto a pelo menos um **servidor de registro**; este conhece o endereço IP do usuário sendo chamado.

Quando Alice precisa se comunicar com Bob, ela pode usar o endereço de e-mail em vez do endereço IP na mensagem de INVITE. A mensagem vai para um servidor *proxy**, que envia uma mensagem de busca (que não faz parte do SIP) para algum servidor de registro que tenha registrado Bob. Quando o servidor *proxy* recebe uma mensagem de resposta do servidor de registro, ele pega a mensagem de INVITE de Alice e insere o recém-descoberto endereço IP de Bob. Essa mensagem é, então, enviada para Bob. A Figura 8.41 ilustra o processo.

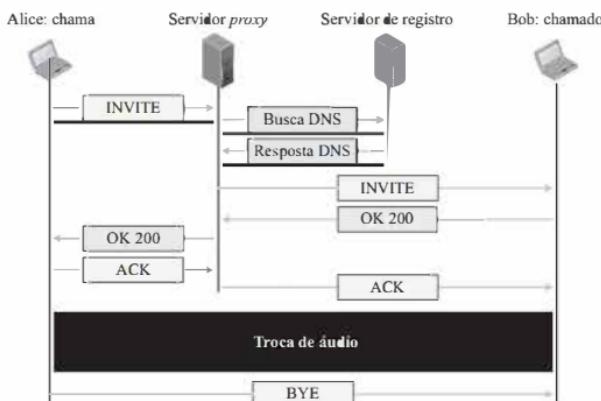



Figura 8.41 Localizando a pessoa chamada.

Formato da mensagem SIP e protocolo SDP

Conforme discutimos anteriormente, as mensagens de pedido e de resposta SIP são divididas em quatro seções: linha inicial ou de estado, cabeçalho, uma linha em branco e corpo. Como uma linha em branco não precisa de informações adicionais, descreveremos brevemente o formato das outras seções.

* N. de T.: Um servidor *proxy* é um servidor que atua como um intermediário na comunicação entre dois computadores.

Linha inicial

A linha inicial consiste em uma única linha que se inicia com o nome da mensagem de pedido, seguido pelo endereço do destinatário e a versão do SIP. Por exemplo, a linha inicial de uma mensagem de pedido do tipo INVITE tem o seguinte formato:

INVITE **sip:forouzan@papaleguas.com**

Linha de estado

A linha de estado consiste em uma única linha que começa com o código de resposta de três dígitos. Por exemplo, a linha de estado de uma mensagem de resposta do tipo 200 tem o seguinte formato.

200 OK

Cabeçalho

Um cabeçalho, na mensagem de pedido ou de resposta, pode usar várias linhas. Cada linha começa com o nome de linha seguido por um sinal de dois pontos e um espaço e depois pelo seu valor. Algumas linhas de cabeçalho típicas são: *Via*, *From*, *To*, *Call-ID*, *Content-Type*, *Content-Length* e *Expired*. O cabeçalho *Via* define o dispositivo SIP por meio do qual a mensagem passa, incluindo o remetente. O cabeçalho *From* (De) define o emissor e o cabeçalho *To* (Para) define o receptor. O cabeçalho *Call-ID* (Identificador da chamada) é um número aleatório que define a sessão. O cabeçalho *Content-Type* (Tipo de conteúdo) define o tipo do corpo da mensagem, que normalmente é SPD, conforme descreveremos em breve. O cabeçalho *Content-Length* (Comprimento do conteúdo) define o comprimento do corpo da mensagem, em *bytes*. O cabeçalho *Expired* (Expirado) é normalmente usado em uma mensagem de registro (REGISTER) para indicar a expiração da informação no corpo. A seguir, mostramos um exemplo de um cabeçalho de uma mensagem do tipo INVITE.

Via: SIP/2.0/UDP 145.23.76.80
From: sip:alice@papaleguas.com
To: sip:bob@pontadeflecha.net
Call-ID: 23a345@ papaleguas.com
Content-Type: application/spd
Content-Length: 600

Corpo

O corpo da mensagem é a principal diferença que veremos entre uma aplicação, como o HTTP e o SIP. O SIP usa outro protocolo, conhecido como Protocolo de Descrição da Sessão (SDP – Session Description Protocol), para definir o corpo. Cada linha do corpo é composta por um código SDP seguido por um sinal de igual, o qual é seguido por um valor. O código é um único caractere que determina a finalidade do código. Podemos dividir o corpo em várias seções. A primeira parte do corpo normalmente contém informações gerais. Os códigos utilizados nessa seção são: ***v*** (versão do SDP) e ***o*** (origem da mensagem).

A segunda parte do corpo normalmente fornece informações ao destinatário para que ele possa decidir fazer parte da sessão. Os códigos utilizados nessa seção são: ***s*** (assunto, ou *subject* em inglês), ***i*** (informações sobre o assunto), ***u*** (URL da sessão) e ***e*** (o endereço de e-mail da pessoa responsável pela sessão).

A terceira parte do corpo fornece os detalhes técnicos para tornar a sessão possível. Os códigos utilizados nessa parte são: ***c*** (o endereço IP *unicast* ou *multicast* que o usuário precisa contatar para ser capaz de fazer parte da sessão), ***t*** (o tempo de início e término da sessão, codifi-

cados como números inteiros), ***m*** (as informações sobre a mídia, como áudio, vídeo, número de porta e protocolo usado).

A seguir, mostramos um exemplo de corpo de uma mensagem de pedido do tipo INVITE.

v = 0
o = forouzan 64.23.45.8
s = aulas de redes
i = o que oferecer no semestre seguinte
u = http://www.uni.edu
e = forouzan@papaleguas.com
c = IN IP4 64.23.45.8
t = 2923721854 2923725454

Juntando as partes

Juntemos as quatro seções de uma mensagem de pedido, conforme mostrado a seguir. A primeira linha é a inicial, enquanto as seis linhas seguintes formam o cabeçalho. A linha seguinte (linha em branco) separa o cabeçalho do corpo da mensagem e as oito últimas linhas compõem o corpo. Concluímos nossa discussão sobre o protocolo SIP e o protocolo auxiliar SPD utilizado pelo SIP para definir o corpo.

INVITE **sip:forouzan@papaleguas.com**
Via: SIP/2.0/UDP 145.23.76.80
From: sip:alice@ papaleguas.com
To: sip:bob@pontadeflecha.net
Call-ID: 23a345@papaleguas.com
Content-Type: application/spd
Content-Length: 600

// Linha em branco

v = 0
o = forouzan 64.23.45.8
s = aulas de redes
i = o que oferecer no semestre seguinte
u = http://www.uni.edu
e = forouzan@ papaleguas.com
c = IN IP4 64.23.45.8
t = 2923721854 2923725454

8.4.5 H.323

O **H.323** é um padrão projetado pela ITU para permitir que telefones da rede de telefonia pública possam se comunicar com computadores (denominados *terminais* no H.323) conectados à Internet. A Figura 8.42 mostra a arquitetura geral do H.323 para áudio, mas ele também pode ser usado para vídeo.

Um ***gateway*** ou **porta de saída** conecta a Internet à rede de telefonia. Em geral, um *gateway* é um dispositivo de cinco camadas capaz de traduzir uma mensagem de uma pilha de protocolos para outra. Aqui, o *gateway* faz exatamente a mesma coisa: transforma uma mensagem da rede de telefonia em uma mensagem da Internet. O ***gatekeeper*** (também conhecido como **guardião**) na

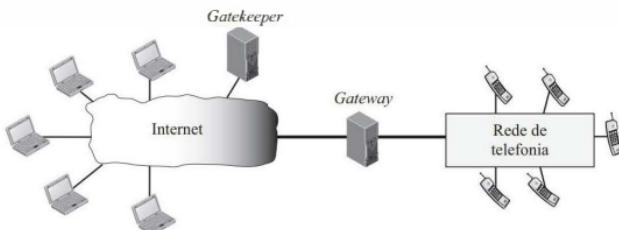



Figura 8.42 Arquitetura do H.323.

rede local é um servidor que desempenha o papel do servidor de registro, conforme discutimos no protocolo SIP.

Protocolos

O H.323 usa uma série de protocolos para estabelecer e manter comunicações de voz (ou vídeo). A Figura 8.43 mostra esses protocolos. O H.323 usa o G.71 ou o G.723.1 para fazer compressão. Ele usa um protocolo denominado H.245, o qual permite que as partes negociem o método de compressão. O protocolo Q.931 é usado para estabelecer e encerrar conexões. Outro protocolo, denominado H.225, ou Registro/Administração/Estado (RAS – Registration/Administration/Status), é utilizado para registro junto ao *gatekeeper*.

Áudio			Controle e sinalização	
Código de compressão / RTP	RTCP	H.225	Q.931	H.245
UDP			TCP	

Figura 8.43 Protocolos H.323.

Precisamos mencionar que o H.323 é um conjunto completo de protocolos que não pode ser comparado com o SIP, que é apenas um protocolo de sinalização, normalmente combinado com o RTP e o RTCP para criar um conjunto completo de protocolos para aplicações de multimídia interativa em tempo real, mas também pode ser usado com outros protocolos. O H.323, por outro lado, é um conjunto completo de protocolos que obriga o uso de RTP e RTCP.

Funcionamento

Daremos um exemplo simples para mostrar a operação de uma comunicação telefônica usando o H.323. A Figura 8.44 mostra os passos executados por um terminal para se comunicar com um telefone.

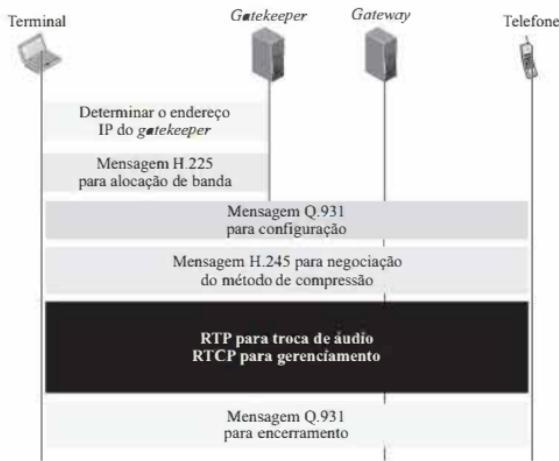



Figura 8.44 Exemplo de H.323.

1. O terminal envia uma mensagem de *broadcast* para o *gatekeeper*, que responde com seu endereço IP.
2. O terminal e o *gatekeeper* se comunicam, usando o H.225 para negociar a largura de banda.
3. O terminal, o *gatekeeper*, o *gateway* e o telefone se comunicam usando o Q.931 para configurar uma conexão.
4. O terminal, o *gatekeeper*, o *gateway* e o telefone se comunicam usando o H.245 para negociar o método de compressão.
5. O terminal, o *gateway* e o telefone trocam dados de áudio usando o RTP sob o gerenciamento do RTCP.
6. O terminal, o *gatekeeper*, o *gateway* e o telefone se comunicam usando o Q.931 para encerrar a comunicação.

8.4.6 SCTP

O **Protocolo de Controle de Fluxo de Transmissão** (SCTP – Stream Control Transmission Protocol) é um novo protocolo da camada de transporte, projetado para combinar alguns recursos do UDP e do TCP em um esforço para criar um protocolo melhor para comunicações multimídia.

Serviços do SCTP

Antes de discutirmos o funcionamento da SCTP, explicaremos os serviços oferecidos pelo SCTP aos processos da camada de aplicação.

Comunicação processo a processo

O SCTP, assim como o UDP e o TCP, fornece comunicação processo a processo.

Múltiplos fluxos

Aprendemos que o TCP é um protocolo orientado a fluxo. Cada conexão entre um cliente TCP e um servidor TCP envolve um único fluxo. O problema com essa abordagem é que a perda de um ponto qualquer no fluxo bloqueia a entrega do restante dos dados. Isto pode ser aceitável quando estamos transferindo texto, mas não quando estamos enviando dados em tempo real, como áudio ou vídeo. O SCTP suporta um **serviço de múltiplos fluxos** em cada conexão, que é denominada uma **associação** na terminologia do SCTP. Se um dos fluxos é bloqueado, os outros ainda podem entregar seus dados. A Figura 8.45 ilustra a ideia da entrega de múltiplos fluxos.

Figura 8.45 Conceito de múltiplos fluxos.

Serviço multiprovedor

Uma conexão TCP envolve um endereço IP de origem e um de destino. Isto significa que mesmo se o emissor ou o receptor for uma estação **multiprovedor** ou ***multihoming*** (ou seja, conectada a mais de um endereço físico com múltiplos endereços IP), apenas um desses endereços IP por ponta pode ser utilizado durante a conexão. Uma associação SCTP, por outro lado, suporta um serviço multiprovedor. As estações emissora e receptora podem definir diversos endereços IP em cada ponta de uma associação. Nessa abordagem tolerante a falhas, quando um caminho falhar, outra interface pode ser usada para propiciar a entrega de dados sem interrupção. Esse mecanismo de tolerância a falhas é muito útil quando estamos enviando e recebendo dados em tempo real, como ocorre na telefonia via Internet. A Figura 8.46 ilustra a ideia do serviço multiprovedor.

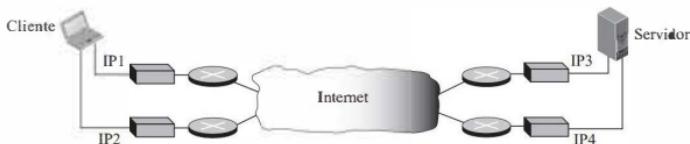



Figura 8.46 Conceito de serviço multiprovedor.

Na Figura, o cliente está conectado a duas redes locais, com dois endereços IP. O servidor também está ligado a duas redes com dois endereços IP. O cliente e o servidor podem criar uma associação utilizando quatro pares de endereços IP diferentes. No entanto, perceba que nas implementações atuais do SCTP, apenas um par de endereços IP pode ser selecionado para a comunicação normal; o par alternativo é utilizado se a escolha principal falhar. Em outras palavras, atualmente, o SCTP não permite o balanceamento de carga entre os diferentes caminhos.

Comunicação *full-duplex*

Assim como o TCP, o SCTP oferece um serviço *full-duplex*, no qual os dados podem fluir em ambas as direções ao mesmo tempo. Cada SCTP tem, então, um *buffer* de envio e um de recepção, sendo que os pacotes são enviados em ambas as direções.

Serviço orientado à conexão

Assim como o TCP, o SCTP é um protocolo orientado à conexão. No entanto, no SCTP, uma conexão é denominada uma *associação*.

Serviço confiável

O SCTP, assim como o TCP, é um protocolo de transporte confiável. Ele usa um mecanismo de confirmação para verificar a chegada de dados sãos e salvos. Discutimos esse recurso mais a fundo na seção sobre controle de erros.

Características do SCTP

A seguir, são mostradas as características gerais do SCTP.

Número de Sequência da Transmissão

A unidade de dados no SCTP é um bloco de dados, que pode ou não ter uma relação um para um com a mensagem proveniente do processo devido à fragmentação (discutida mais tarde). A transferência de dados no SCTP é controlada por meio da numeração dos blocos de dados. O SCTP usa um *Número de Sequência da Transmissão* (TSN – Transmission Sequence Number) para numerar os blocos de dados. Em outras palavras, o TSN no SCTP desempenha um papel análogo ao número de sequência no TCP. Os TSNs têm 32 *bits* de comprimento e são inicializados com um valor aleatório entre 0 e $2^{32} - 1$. Cada bloco de dados deve carregar o TSN correspondente em seu cabeçalho.

Identificador de Fluxo

No SCTP, pode haver vários fluxos em cada associação. Cada fluxo no SCTP precisa ser identificado usando um *Identificador de Fluxo* (SI – Stream Identifier). Cada bloco de dados deve carregar o SI em seu cabeçalho para que, quando chegar ao destino, ele possa ser devidamente inserido em seu fluxo. O SI é um número de 16 *bits* começando em 0.

Número de Sequência do Fluxo

Quando um bloco de dados chega ao SCTP de destino, ele é entregue ao fluxo adequado e na ordem correta. Isto significa que, além de usar um SI, o SCTP identifica cada bloco de dados em cada fluxo usando um *Número de Sequência do Fluxo* (SSN – Stream Sequence Number).

Pacotes

No TCP, um segmento carrega dados e informações de controle. Os dados são transportados como uma coleção de *bytes*; a informação de controle é definida por seis marcadores de controle no cabeçalho. O projeto do SCTP é totalmente diferente: os dados são transportados como blocos de dados e as informações de controle como blocos de controle. Diversos blocos de controle e blocos de dados podem ser empacotados juntos em um pacote. Um pacote no SCTP desempenha o mesmo papel que um segmento no TCP. A Figura 8.47 compara um segmento no TCP com um pacote no SCTP. Discutimos o formato do pacote SCTP na próxima seção.

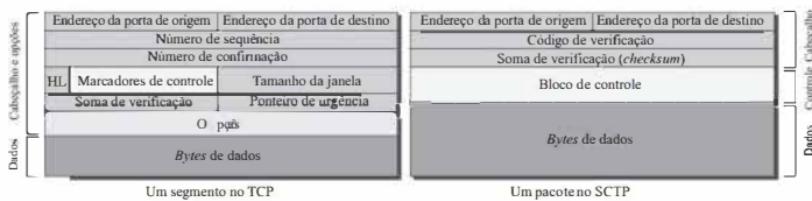



Figura 8.47 Comparação entre um segmento TCP e um pacote SCTP.

No SCTP, temos blocos de dados, fluxos e pacotes. Uma associação pode enviar muitos pacotes, um pacote pode conter vários blocos e os blocos podem pertencer a diferentes fluxos. Para clarificar as definições desses termos, consideremos que o processo A precisa enviar 11 mensagens para o processo B em três fluxos. As primeiras quatro mensagens ficam no primeiro fluxo, as três mensagens seguintes ficam no segundo fluxo e as últimas quatro mensagens ficam no terceiro fluxo. Embora uma mensagem, se for longa, possa ser transportada por vários blocos de dados, consideramos que cada mensagem caiba em um bloco de dados. Portanto, temos 11 blocos de dados em três fluxos.

O processo da camada de aplicação entrega 11 mensagens para o SCTP, onde cada mensagem é marcada como pertencente ao fluxo adequado. Embora o processo possa entregar uma mensagem do primeiro fluxo e depois outra do segundo, consideramos que ela entrega primeiramente todas as mensagens pertencentes ao primeiro fluxo, em seguida todas as pertencentes ao segundo fluxo, e finalmente todas pertencentes ao último fluxo. Consideramos também que a rede permite apenas três blocos de dados por pacote, o que significa que precisamos de quatro pacotes, conforme mostra a Figura 8.48.

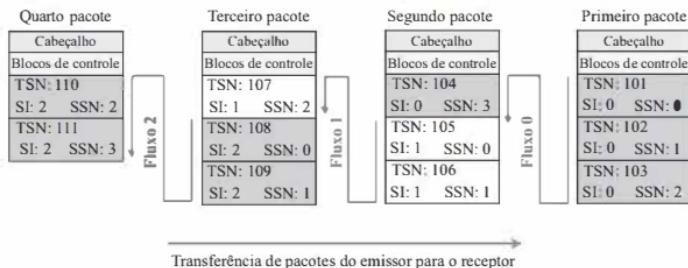



Figura 8.48 Pacotes, blocos de dados e fluxos.

Os blocos de dados no fluxo 0 são transportados no primeiro pacote e em uma parte do segundo; os blocos do fluxo 1 são transportados no segundo e no terceiro pacotes; os blocos do fluxo 2 são transportados no terceiro e quarto pacotes.

Perceba que cada bloco de dados requer três identificadores: TSN, SI e SSN. O TSN é um número cumulativo e é usado, conforme veremos mais adiante, para controle de fluxo e de erros. O SI define o fluxo ao qual o bloco pertence. O SSN define a ordem do bloco em um determinado fluxo. Em nosso exemplo, o SSN começa a partir de 0 em cada fluxo.

Número de confirmação

Os números de confirmação do TCP são orientados a *bytes* e referem-se aos números de sequência. Os números de confirmação do SCTP são orientados a blocos. Eles referem-se aos TSNs. Uma segunda diferença entre as confirmações no TCP e no SCTP são as informações de controle. Lembre-se de que essas informações fazem parte do cabeçalho do segmento no TCP. Para confirmar segmentos que transportam apenas informações de controle, o TCP usa um número de sequência e um de confirmação (por exemplo, um segmento de SYN precisa ser confirmado por um segmento de ACK). No SCTP, entretanto, a informação de controle é transportada por blocos de controle, que não precisam de um TSN. Esses blocos de controle são confirmados por outro bloco de controle do tipo apropriado (alguns não precisam ser confirmados). Por exemplo, a recepção de um bloco de controle do tipo INIT é confirmada por um bloco de INIT-ACK. Não é necessário um número de sequência ou um número de confirmação.

Formato dos pacotes

Um pacote SCTP tem um cabeçalho geral obrigatório e um conjunto de blocos, também conhecidos como *chunks* ou pedaços. Existem dois tipos de blocos: blocos de controle e blocos de dados. Um bloco de controle controla e mantém a associação; um bloco de dados transporta os dados do usuário. Em um pacote, os blocos de controle vêm antes dos blocos de dados. A Figura 8.49 mostra o formato geral de um pacote SCTP.

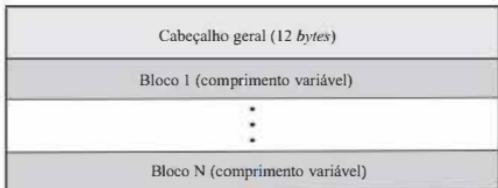



Figura 8.49 Formato do pacote SCTP.

Cabeçalho geral

O *cabeçalho geral* (cabeçalho do pacote) define as pontas de cada associação à qual o pacote pertence, garante que o pacote pertence a uma associação em particular e preserva a integridade do seu conteúdo, incluindo o cabeçalho em si. O formato do cabeçalho geral é mostrado na Figura 8.50.

Endereço da porta de origem 16 *bits*	Endereço da porta de destino 16 *bits*
Código de verificação 32 *bits*	
Soma de verificação 32 *bits*	

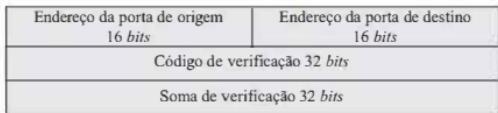



Figura 8.50 Cabeçalho geral.

Há quatro campos no cabeçalho geral. Os números de porta de origem e de destino são equivalentes àqueles do UDP ou do TCP. O código de verificação é um campo de 32 *bits* que associa um

pacote a uma associação. Isto impede que um pacote de uma associação anterior seja confundido com um pacote desta associação. Ele serve como um identificador para a associação, sendo repetido em todos os pacotes ao longo dela. O campo seguinte é uma soma de verificação (*checksum*). No entanto, o tamanho da soma de verificação é expandido de 16 *bits* (no UDP, TCP e IP) para 32 *bits* no SCTP para permitir o uso de uma soma de verificação do tipo CRC-32.

Blocos

Informações de controle e dados de usuário são transportados em blocos, que apresentam uma disposição em comum, conforme mostra a Figura 8.51. Os três primeiros campos são comuns a todos os blocos; o campo de informação depende do tipo de bloco. O campo de tipo pode definir até 256 tipos de blocos. Apenas alguns foram definidos até agora; os tipos restantes são reservados para uso futuro. O campo de marcadores contém marcadores especiais que podem ser necessários por um bloco específico. Cada *bit* tem um significado diferente dependendo do tipo de bloco. O campo de comprimento define o tamanho total do bloco, em *bytes*, incluindo os campos de tipo, de marcadores e de comprimento. Como o tamanho da seção de informações depende do tipo do bloco, é necessário definir os limites. Se um bloco não carrega qualquer informação, o valor do campo de comprimento é 4 (4 *bytes*). Perceba que o comprimento dos *bytes* de enchimento (*padding*), caso esteja presente, não é incluído no cálculo do campo de comprimento. Isso ajuda o receptor a descobrir quantos *bytes* úteis um bloco carrega. Se o valor não for um múltiplo de 4, o receptor sabe que há *bytes* de enchimento.

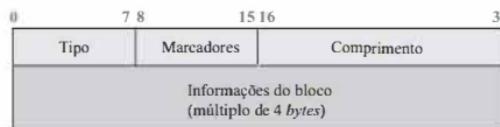



Figura 8.51 Disposição em comum de um bloco.

Tipos de blocos O SCTP define diversos tipos de blocos, conforme mostra a Tabela 8.8.

Tabela 8.8 Blocos.

Tipo	Bloco	Descrição
0	DATA	Dados de usuário
1	INIT	Configura uma associação
2	INIT-ACK	Confirma um bloco de INIT
3	SACK	Confirmação seletiva
4	HEARTBEAT	Verifica se um usuário continua ativo
5	HEARTBEAT ACK	Confirma o bloco de HEARTBEAT
6	ABORT	Aborta uma associação
7	SHUTDOWN	Encerra uma associação

(Continua)

Tabela 8.8 Blocos (Continuação)		
Tipo	Bloco	Descrição
8	SHUTDOWN ACK	Confirma um bloco de SHUTDOWN
9	ERROR	Relata erros sem encerrar a associação
10	COOKIE ECHO	Terceiro pacote no estabelecimento de uma associação
11	COOKIE ACK	Confirma um bloco de COOKIE ECHO
14	SHUTDOWN COMPLETE	Terceiro pacote no encerramento de uma associação
192	FORWARD TSN	Para ajustar um TSN cumulativo

Uma associação SCTP

O SCTP, assim como o TCP, é um protocolo orientado à conexão. No entanto, uma conexão no SCTP é denominada uma *associação* para enfatizar o serviço multiprovedor.

Uma conexão no SCTP é denominada uma associação.

Estabelecimento de associação

Um estabelecimento de associação no SCTP requer um procedimento de *four-way handshaking*, ou *apresentação em quatro vias*. Nesse procedimento, um processo, normalmente um cliente, deseja estabelecer uma associação com outro processo, normalmente um servidor, usando o SCTP como o protocolo da camada de transporte. De forma similar ao TCP, o servidor SCTP precisa estar preparado para receber qualquer associação (abertura passiva). O estabelecimento da associação, no entanto, é iniciado pelo cliente (abertura ativa). O estabelecimento da associação SCTP é mostrado na Figura 8.52.

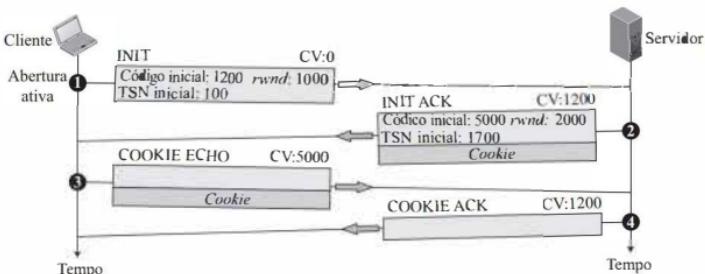



Figura 8.52 Apresentação em quatro vias (*four-way handshaking*).

Os passos do processo, em uma situação normal, são os seguintes:

1. O cliente envia o primeiro pacote, que contém um bloco de INIT. O *código de verificação* (CV) desse pacote (definido no cabeçalho geral) é 0 porque nenhum código de verificação foi ainda definido para essa direção (do cliente para o servidor). O bloco de INIT inclui um código inicial para ser usado por pacotes indo na direção oposta (do servidor para o cliente). O bloco também define o TSN inicial para essa direção e anuncia um valor para *rwnd*. O valor de *rwnd* é normalmente anunciado em um bloco de SACK; isto é feito aqui porque o SCTP permite a inclusão de um bloco de DADOS no terceiro e quarto pacotes; o servidor deve estar ciente do tamanho disponível no *buffer* do cliente. Perceba que nenhum outro bloco pode ser enviado com o primeiro pacote.
2. O servidor envia o segundo pacote, que contém um bloco de INIT ACK. O código de verificação é o valor do campo de código inicial no bloco de INIT. Esse bloco inicia o código a ser utilizado na outra direção, definindo o TSN inicial para os dados que estão indo do servidor para o cliente e também o *rwnd* do servidor. O valor de *rwnd* é definido para permitir que o cliente envie um bloco de DATA com o terceiro pacote. Um INIT ACK também carrega um *cookie* que indica o estado do servidor nesse momento. Discutiremos o uso do *cookie* em breve.
3. O cliente envia o terceiro pacote, que inclui um bloco de COOKIE ECHO, muito simples, que ecoa, sem alterações, o *cookie* enviado pelo servidor. O SCTP permite a inclusão de blocos de dados nesse pacote.
4. O servidor envia o quarto pacote, que inclui um bloco de COOKIE ACK para confirmar o recebimento do bloco de COOKIE ECHO. O SCTP permite a inclusão de blocos de dados com esse pacote.

Transferência de dados

A finalidade de uma associação é transferir dados entre duas pontas da comunicação. Após a associação ser estabelecida, a transferência de dados bidirecional pode ser feita. O cliente e o servidor podem ambos enviar dados. Assim como o TCP, o SCTP suporta mecanismos de carona (*piggybacking*).

Há uma grande diferença, entretanto, entre a transferência de dados no TCP e no SCTP. O TCP recebe mensagens de um processo na forma de um fluxo de *bytes* ignorando qualquer delimitação entre eles. O processo pode inserir delimitadores para utilização pelo seu par na camada de aplicação, mas o TCP trata tais delimitadores como parte do texto. Em outras palavras, o TCP recebe cada mensagem e a adiciona ao seu *buffer*. Um segmento pode carregar partes de duas mensagens deferentes. O único sistema de ordenação imposto pelo TCP refere-se aos números dos *bytes*.

O SCTP, por outro lado, reconhece e preserva fronteiras entre mensagens. Cada mensagem proveniente do processo é tratada como uma unidade e inserida em um bloco de dados (DATA) exceto no caso de fragmentação (discutido mais adiante). Nesse sentido, o SCTP é como o UDP, com uma grande vantagem: existe uma relação entre os blocos de dados.

Uma mensagem recebida de um processo torna-se um bloco de DATA, ou blocos, caso seja fragmentada, adicionando-se um cabeçalho de blocos do tipo DATA à mensagem. Cada bloco de DATA formado por uma mensagem ou um fragmento de uma mensagem recebe um TSN. Precisamos lembrar que apenas blocos do tipo DATA usam TSNs e que apenas blocos do tipo DATA têm sua recepção confirmada por blocos de SACK.

Transferência de dados multiprovedor Discutimos o suporte ao serviço multiprovedor ou *multihoming* do SCTP, uma característica que distingue o SCTP do UDP e do TCP. O serviço multiprovedor permite que ambas as pontas da comunicação definam diversos endereços IP para a comunicação. No entanto, apenas um desses endereços pode ser definido como o *endereço primário*; o restante são endereços alternativos. O endereço primário é definido durante o estabelecimento da associação. O ponto interessante é que o endereço primário de uma ponta é determinado pela outra ponta. Em outras palavras, uma origem define o endereço primário usado por um destino.

A transferência de dados, por padrão, usa o endereço primário do destino. Se o endereço primário não estiver disponível, um dos endereços alternativos é usado. O processo, no entanto, sempre pode substituir o endereço primário e explicitamente solicitar que uma mensagem seja enviada para um dos endereços alternativos. Um processo pode também alterar explicitamente o endereço primário da associação atual.

Uma pergunta lógica que surge é para onde enviar uma mensagem de SACK. O SCTP dita que um SACK deve ser enviado para o endereço de origem do pacote SCTP correspondente.

Entrega multiprovedor Uma característica interessante do SCTP é a distinção entre transferência e entrega de dados. O SCTP usa números TSN para lidar com a transferência de dados, o movimento de blocos de dados entre a origem e o destino. A entrega dos blocos de dados é controlada por SIs e SSNs. O SCTP é capaz de suportar múltiplos fluxos contínuos, o que significa que o processo emissor pode definir diferentes fluxos e uma mensagem pode pertencer a um deles. A cada fluxo é atribuído um identificador de fluxo (SI), que define univocamente aquele fluxo. No entanto, o SCTP suporta dois tipos de entrega de dados em cada fluxo: *ordenada* (padrão) e *não ordenada*. Na entrega ordenada de dados, blocos de dados em um fluxo usam Números de Sequência do Fluxo (SSNs – Stream Sequence Numbers) para definir sua ordem no fluxo. Quando os blocos chegam ao destino, o SCTP é responsável pela entrega das mensagens de acordo com o SSN definido no bloco. Isso pode atrasar a entrega, pois alguns blocos podem chegar fora de ordem. Na entrega *não ordenada* de dados, os blocos de dados em um fluxo têm ativado seu marcador *U* (de *unnordered*, ou não ordenado), de modo que o valor do campo SSN é ignorado. Eles não consomem SSNs. Quando um bloco de dados não ordenado chega ao SCTP de destino, ele entrega a mensagem que transporta o bloco para a camada de aplicação sem esperar pelas outras mensagens. Na maioria das vezes, os aplicativos usam o serviço de entrega ordenada, porém ocasionalmente alguns aplicativos precisam enviar dados urgentes que devem ser entregues fora de ordem (lembre-se dos dados urgentes e do recurso de ponteiro urgente do TCP). Nesses casos, o aplicativo pode especificar que a entrega deve ser não ordenada.

Fragmentação Outra questão na transferência de dados é a *fragmentação*. Embora o SCTP compartilhe esse termo com o IP (ver Capítulo 4), as fragmentações no IP e no SCTP pertencem a níveis diferentes: o primeiro refere-se à camada de rede, enquanto o último à camada de transporte.

O SCTP preserva as fronteiras da mensagem quando cria um bloco de DATA a partir dela caso seu tamanho (quando encapsulado em um datagrama IP) não exceda a MTU (ver Capítulo 4) do caminho. O tamanho de um datagrama IP transportando uma mensagem pode ser determinado somando-se o tamanho da mensagem, em *bytes*, aos tamanhos de quatro conjuntos de dados de controle: cabeçalho do bloco de dados, blocos de SACK necessários, cabeçalho geral do SCTP e cabeçalho IP. Se o tamanho total exceder o valor da MTU, a mensagem precisa ser fragmentada. A fragmentação no SCTP de origem segue os passos a seguir:

1. A mensagem é dividida em fragmentos menores para satisfazer o requisito de tamanho.
2. Um cabeçalho de bloco de dados (DATA) é adicionado a cada fragmento que carrega um TSN diferente. Os TSNs precisam estar em sequência.
3. Todos os cabeçalhos dos blocos carregam o mesmo Identificador de Fluxo (SI – Stream Identifier), o mesmo Número de Sequência do Fluxo (SSN – Stream Sequence Number), o mesmo identificador do protocolo da carga útil e o mesmo valor para o marcador *U*.
4. A combinação dos marcadores *B* e *E* são as seguintes*:

    a. Primeiro fragmento: 10.

    b. Fragmentos intermediários: 00.

    c. Último fragmento: 01.

* N. de T.: No SCTP, o marcador B (de *beginning*, ou início), indica o fragmento inicial. Já o marcador E (de *end*, ou fim) indica o fragmento final.

Os fragmentos são reagrupados no destino. Se um bloco de DATA chegar com os *bits* B/E iguais a 1/1, significa que ele não foi fragmentado. O receptor sabe como reagrupar todos os blocos com os mesmos SIs e SSNs. O número de fragmentos é determinado pelo TSN do primeiro e do último fragmento.

Encerramento da associação

No SCTP, assim como no TCP, qualquer uma das duas partes envolvidas na troca de dados (cliente ou servidor) pode encerrar a conexão. No entanto, ao contrário do TCP, o SCTP não permite uma associação "semifechada". Se uma das pontas fecha a associação, a outra ponta deve parar de enviar novos dados. Se quaisquer dados forem deixados na fila do destinatário do pedido de encerramento, eles são enviados e a associação é fechada. O encerramento da associação utiliza três pacotes, conforme mostra a Figura 8.53. Perceba que, embora a figura mostre o caso em que o encerramento é iniciado pelo cliente, ele pode também ser iniciado pelo servidor.

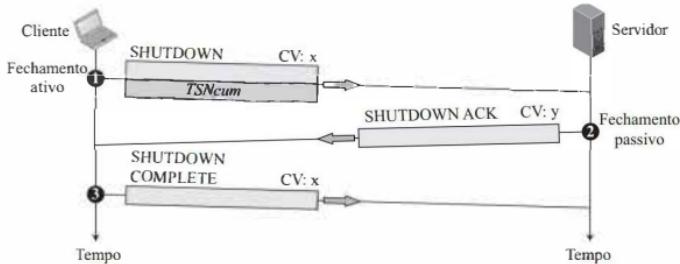



Figura 8.53 Encerramento da associação.

Controle de fluxo

O controle de fluxo no SCTP é semelhante àquele do TCP. Neste, precisamos lidar com apenas uma unidade de dados, o *byte*. No SCTP, precisamos lidar com duas unidades de dados, o *byte* e o bloco. Os valores de *rwnd* e *cwnd* são expressados em *bytes*; os valores do TSN e das confirmações são expressados em blocos. Para ilustrar o conceito, faremos algumas suposições não realistas. Consideramos que nunca há congestionamento na rede e que a rede é livre de erros. Em outras palavras, assumimos que o valor de *cwnd* é infinito e que nenhum pacote é perdido ou atrasado, nem chega fora de ordem, e também que a transferência de dados é unidirecional. Corrigiremos nossas considerações não realistas em seções posteriores. As implementações atuais do SCTP ainda usam uma janela orientada a *bytes* para o controle de fluxo. No entanto, mostramos um *buffer* em termos de blocos para facilitar a compreensão do conceito.

Lado do receptor

O receptor tem um *buffer* (fila) e três variáveis. A fila contém os blocos de dados recebidos que ainda não foram lidos pelo processo. A primeira variável, *TSNcum* (TSN cumulativo), contém o valor do último TSN recebido. A segunda variável, *tamJan*, indica o tamanho disponível do *buffer*. A terceira variável, *ultACK*, guarda o valor da última confirmação cumulativa. A Figura 8.54 mostra a fila e as variáveis no lado do receptor.

1. Quando um lado recebe um bloco de dados, ele o armazena no final do *buffer* (fila) e subtrai o tamanho do bloco de *tamJan*. O número de TSN do bloco é armazenado na variável *TSNcum*.

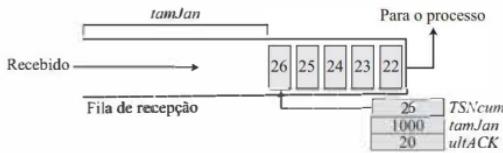



Figura 8.54 Controle de fluxo no lado do receptor.

2. Quando o processo lê um bloco, ele remove aquele bloco da fila e adiciona o tamanho do bloco removido à variável *tamJan* (reciclagem).
3. Quando o receptor decide enviar um SACK, ele verifica o valor de *ultAck*; se for inferior ao valor de *TSNcum*, ele envia um SACK com um número de TSN cumulativo igual ao valor de *TSNcum*. Também inclui o valor de *tamJan* como o tamanho da janela anunciado. O valor da *ultACK* é então atualizado para o valor de *TSNcum*.

Lado do emissor

O emissor tem um *buffer* (fila) e três variáveis: *TSNatual*, *rwnd*, e *emTrânsito*, conforme mostra a Figura 8.55. Consideramos que o tamanho de cada bloco é de 100 *bytes*.

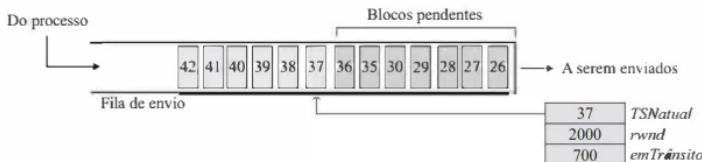



Figura 8.55 Controle de fluxo no lado do emissor.

O *buffer* armazena os blocos produzidos pelo processo que tenham sido enviados ou estejam prontos para serem enviados. A primeira variável, *TSNatual*, indica o próximo bloco a ser enviado. Todos os blocos na fila com um TSN inferior a esse valor foram enviados, mas não confirmados; eles estão pendentes. A segunda variável, *rwnd*, contém o último valor anunciado pelo receptor (em *bytes*) para o tamanho da janela. A terceira variável, *emTrânsito*, registra o número de *bytes* em trânsito, os *bytes* que foram enviados, mas cuja recepção ainda não foi confirmada. O procedimento utilizado pelo emissor é descrito a seguir.

1. Um bloco apontado por *TSNatual* pode ser enviado se o tamanho dos dados for menor do que ou igual ao valor de (*rwnd* – *emTrânsito*). Após o envio do bloco, o valor de *TSNatual* é incrementado de 1 e passa, então, a apontar para o próximo bloco a ser enviado. O valor de *emTrânsito* é incrementado de um valor igual ao tamanho dos dados no bloco transmitido.
2. Quando um SACK é recebido, os blocos com um TSN inferior ou igual ao valor do TSN cumulativo no SACK são removidos da fila e descartados. O emissor não precisa mais se preocupar com eles. O valor da variável *emTrânsito* é decrementado do tamanho total dos blocos descartados. O valor de *rwnd* é atualizado para o valor da janela anunciado no SACK.

Controle de erros

O SCTP, assim como o TCP, é um protocolo confiável da camada de transporte. Ele usa um bloco de SACK para informar o estado do *buffer* de recepção para o emissor. Cada implementação usa um conjunto diferente de entidades e temporizadores para os lados do receptor e do emissor. Usamos um projeto muito simples para transmitir o conceito para o leitor.

Lado do receptor

Em nosso projeto, o receptor armazena na fila todos os blocos que chegaram, inclusive aqueles fora de ordem. No entanto, ele deixa espaços vazios para quaisquer pedaços faltantes. Descarta mensagens duplicadas, mas mantém um registro delas para gerar relatórios para o emissor. A Figura 8.56 mostra um projeto típico para o lado do receptor e o estado da fila de recepção em um instante em particular.

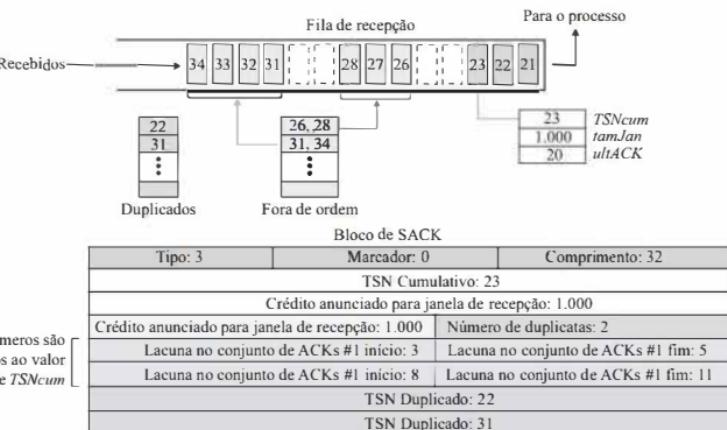



Figura 8.56 Controle de erros no lado do receptor.

A última confirmação foi enviada para o bloco de dados de número 20. O tamanho da janela disponível é de 1.000 *bytes*. Os blocos 21 a 23 foram recebidas em ordem. O primeiro conjunto fora de ordem contém os blocos de 26 a 28. O segundo conjunto fora de ordem contém os blocos de 31 a 34. Uma variável armazena o valor de *TSNcum*. Uma matriz de variáveis mantém um registro do início e do final de cada conjunto de blocos que estejam fora de ordem. Uma matriz de variáveis registra os blocos duplicados recebidos. Perceba que não é necessário armazenar blocos duplicados na fila, pois eles são descartados. A figura também mostra o bloco de SACK que será enviado para informar o estado do receptor para o emissor. Os números de TSN para os blocos fora de ordem são relativos (deslocamentos) ao TSN cumulativo.

Lado do emissor

No lado do emissor, nosso projeto requer dois *buffers* (filas): uma fila de envio e uma de retransmissão. Também usamos três variáveis: *rwnd*, *emTrânsito* e *TSNatual*, conforme descrito na seção anterior. A Figura 8.57 mostra um projeto típico.

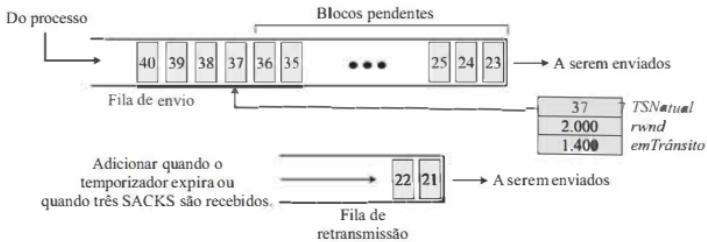



Figura 8.57 Controle de erros no lado emissor.

A fila de envio armazena os blocos de 23 a 40. Os blocos de 23 a 36 já foram enviados, mas sua recepção não foi confirmada; são blocos pendentes. O valor de *TSNatual* aponta para o próximo bloco a ser enviado (37). Consideramos que cada bloco tem 100 *bytes*, o que significa que 1.400 *bytes* de dados (blocos de 23 a 36) encontram-se em trânsito. O emissor, nesse momento, tem uma fila de retransmissão. Quando um pacote é enviado, um temporizador de retransmissão é iniciado para esse pacote. Algumas implementações usam um temporizador único para a associação toda, mas continuamos com nossa tradição de usar um temporizador para cada pacote por razões de simplicidade. Quando o temporizador de retransmissão de um pacote expira, ou quando três SACKs chegam declarando um pacote como ausente (a retransmissão rápida foi discutida para o TCP), os blocos daquele pacotes são movidos para a fila de retransmissão para serem reenviados. Esses blocos são considerados perdidos, e não pendentes. Os blocos na fila de retransmissão têm maior prioridade. Em outras palavras, a próxima vez que o emissor enviar um bloco, isto será feito para o bloco 21 da fila de retransmissão.

Para ver como o estado do emissor muda, suponha que o SACK na Figura 8.56 chega ao local do emissor da Figura 8.57. A Figura 8.58 mostra o novo estado.

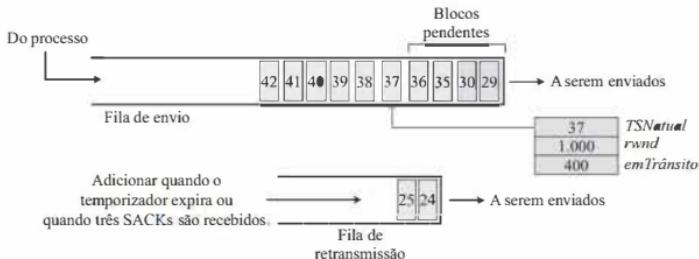



Figura 8.58 Novo estado no lado emissor após a recepção de um bloco de SACK.

1. Todos os blocos com um TSN igual ou inferior ao valor de *TSNcum* no SACK são removidos da fila de envio ou de retransmissão. Eles não estão mais pendentes ou marcados para retransmissão. Os blocos 21 e 22 são removidos da fila de retransmissão e o bloco 23 é removido da fila de envio.

2. Nosso projeto também remove todos os blocos da fila de envio que são declarados nos conjuntos de lacunas; algumas implementações conservadoras, no entanto, mantêm esses blocos até a recepção de um *TSNcum* que os inclua. Essa precaução é necessária para a rara situação na qual o receptor encontra algum problema com esses blocos fora de ordem. Nosso projeto ignora essas situações raras. Os blocos de 26 a 28 e os blocos de 31 a 34 são, portanto, removidos da fila de envio.
3. A lista de blocos duplicados não sofre qualquer efeito.
4. O valor de *rwnd* é alterado para 1.000 conforme anunciado no bloco de SACK.
5. Também consideramos que o temporizador de transmissão para o pacote que carregava os blocos 24 e 25 expirou. Eles são movidos para a fila de retransmissão e um novo temporizador de retransmissão é iniciado de acordo com a regra de recuo exponencial discutida para o TCP.
6. O valor da variável *emTrânsito* passa a ser 400 porque apenas quatro pedaços estão agora em trânsito. Os blocos na fila de retransmissão não são contados porque se considera que eles foram perdidos, não que estão em trânsito.

Enviando blocos de dados

Uma extremidade da comunicação pode enviar um pacote de dados sempre que houver blocos de dados na sua fila de envio cujo TSN for superior ou igual ao valor de *TSNatual*, ou se houver blocos de dados na fila de retransmissão. A fila de retransmissão tem maior prioridade. No entanto, o tamanho total do bloco ou blocos de dados incluídos no pacote não deve exceder o valor (*rwnd – emTrânsito*) e o tamanho total do quadro não deve exceder o tamanho da MTU, conforme discutimos nas seções anteriores. Se considerarmos, em nosso cenário anterior, que nosso pacote pode carregar três blocos (devido à restrição da MTU), então podem ser enviados os blocos 24 e 25, da fila de retransmissão e o bloco 37 (o próximo bloco da fila de envio) está pronto para ser enviado. Perceba que os blocos pendentes na fila de envio não podem ser enviados, pois supostamente estão em trânsito. Note também que qualquer bloco enviado a partir da fila de retransmissão também é temporizado para a retransmissão novamente. O novo temporizador afeta os blocos 24, 25 e 37. Precisamos mencionar aqui que algumas implementações podem não permitir a mistura de blocos da fila de retransmissão e da fila de envio. Nesse caso, apenas os blocos 24 e 25 podem ser enviados no pacote. O formato do bloco de dados encontra-se na página Web do livro.

Retransmissão Para controlar a perda ou o descarte de um bloco, o SCTP, como o TCP, emprega duas estratégias: o uso de temporizadores de retransmissão e a recepção de três SACKs referentes a um mesmo bloco faltante.

- **Retransmissão.** O SCTP usa um temporizador de retransmissão para tratar o tempo de retransmissão, o tempo de espera por uma confirmação de um segmento. Os procedimentos para o cálculo do RTO e do RTT no SCTP são iguais àqueles descritos para o TCP. O SCTP usa um RTT medido (RTT_M), um RTT suavizado (RTT_S) e um desvio do RTT (RTT_D) para calcular o RTO. O SCTP também usa o algoritmo de Karn para evitar ambiguidade nas confirmações. Perceba que, se uma estação estiver usando mais do que um endereço IP (cenário multiprovedor), RTOs separados devem ser calculados e mantidos para cada caminho.
- **Quatro relatos de perda.** Sempre que um emissor recebe quatro SACKs cuja informação da lacuna de ACKs indica que um ou mais blocos de dados específicos estão faltando, o emissor precisa considerar que esses blocos foram perdidos e deve movê-los imediatamente para a fila de retransmissão. Esse comportamento é análogo à "retransmissão rápida" no TCP.

Gerando blocos de SACK

Outra questão no controle de erros refere-se à geração de blocos de SACK. As regras para a geração de blocos de SACK no SCTP são semelhantes às regras utilizadas para a confirmação com o marcador de ACK do TCP. Resumimos a lista de regras a seguir.

1. Quando uma extremidade da comunicação envia um bloco de dados (DATA) para a outra extremidade, ela deve incluir um bloco de SACK anunciando o recebimento de blocos de dados ainda não confirmados.
2. Quando uma extremidade recebe um pacote contendo dados, mas não tem dados para enviar, ela deve confirmar a recepção do pacote dentro de um tempo especificado (geralmente 500 ms).
3. Uma extremidade deve enviar pelo menos um SACK para cada pacote que recebe. Essa regra tem precedência sobre a segunda regra.
4. Quando chega um pacote com blocos fora de ordem, o receptor precisa enviar imediatamente um bloco de SACK para relatar o ocorrido para o emissor.
5. Quando uma extremidade recebe um pacote com blocos de dados (DATA) duplicados e nenhum novo bloco de dados, ela deve relatar os blocos de dados duplicados imediatamente com um bloco de SACK.

Controle de congestionamento

O SCTP, assim como o TCP, é um protocolo da camada de transporte cujos pacotes estão sujeitos a congestionamento na rede. Os projetistas do SCTP usaram as mesmas estratégias para controle de congestionamento que aquelas empregadas no TCP.

8.5 QUALIDADE DE SERVIÇO

A Internet foi originalmente projetada para serviços de melhor esforço sem garantias previsíveis de desempenho. O serviço de melhor esforço é, muitas vezes, suficiente para um tráfego que não seja sensível a atrasos, como é o caso de transferências de arquivos e e-mail. Esse tráfego é conhecido como *elástico*, pois é maleável o suficiente para funcionar em condições de atraso; também é conhecido como tráfego de *taxa de bits disponível* porque as aplicações podem acelerar ou desacelerar o envio de dados de acordo com a taxa de *bits* disponível.

O tráfego em tempo real, gerado por algumas aplicações multimídia, é sensível a atrasos e requer, portanto, desempenho garantido e previsível.

A **Qualidade de Serviço** (QoS – Quality of Service) é uma questão da interligação de redes que se refere a um conjunto de técnicas e mecanismos que garantem o desempenho da rede, com o objetivo de oferecer um serviço previsível para um programa da camada de aplicação.

8.5.1 Características de fluxos de dados

Se quisermos fornecer qualidade de serviço para uma aplicação da Internet, precisamos primeiro definir o que precisamos para cada aplicação. Tradicionalmente, são atribuídos quatro tipos de características a um fluxo: *confiabilidade*, *atraso*, *jitter* e *largura de banda*. Primeiramente, definiremos essas características e, em seguida, investigaremos os requisitos de cada tipo de aplicação.

Definições

Podemos apresentar definições informais das quatro características anteriores:

- **Confiabilidade.** É uma característica que um fluxo precisa para entregar os pacotes sãos e salvos ao destino. A falta de confiabilidade significa a perda de um pacote ou de uma confirmação, o que implica retransmissão. Entretanto, a sensibilidade de diferentes aplicações à confiabilidade varia. Por exemplo, a transmissão confiável é mais importante para aplicações de correio eletrônico, transferência de arquivos e acesso à Internet do que para aplicações de telefonia ou de audioconferência.

- **Atraso.** O *atraso* da origem até o destino é outra característica dos fluxos. Novamente, as aplicações podem tolerar o atraso em graus diferentes. Nesse caso, aplicações de telefonia, audioconferência, videoconferência e acesso remoto exigem um atraso reduzido, enquanto na transferência de arquivos ou no e-mail, o atraso é menos importante.
- ***Jitter.*** É a variação de atraso para pacotes que pertencem ao mesmo fluxo. Por exemplo, se quatro pacotes forem enviados nos instantes de tempo 0, 1, 2 e 3, e chegarem ao destino nos instantes 20, 21, 22 e 23, todos apresentam o mesmo atraso, de 20 unidades de tempo. Por outro lado, se os quatro pacotes anteriores chegarem ao destino nos instantes 21, 23, 24 e 28, eles têm diferentes atrasos. Para aplicações como áudio e vídeo, o primeiro caso é completamente aceitável; o segundo caso não é. Para essas aplicações, não importa se os pacotes chegam com um atraso curto ou longo, contanto que o atraso seja o mesmo para todos os pacotes. Esses tipos de aplicação não toleram *jitter*.
- **Largura de banda.** Diferentes aplicações exigem diferentes larguras de banda. Em aplicações de videoconferência, precisamos enviar milhões de *bits* por segundo para atualizar uma tela colorida, enquanto o número total de *bits* em um e-mail pode nem sequer atingir um milhão.

Sensibilidade de aplicações

Agora, veremos como diversas aplicações são sensíveis a algumas características do fluxo. A Tabela 8.9 apresenta um resumo de tipos de aplicações e de suas sensibilidades.

Para aplicações com um alto nível de sensibilidade à confiabilidade, precisamos aplicar mecanismos de verificação de erros e descartar pacotes se eles estiverem corrompidos. Para aplicações com um alto nível de sensibilidade ao atraso, precisamos ter certeza de que elas recebam maior prioridade na transmissão. Para as aplicações com um alto nível de sensibilidade ao *jitter*, precisamos ter certeza de que os pacotes pertencentes à mesma aplicação atravessam a rede com o mesmo atraso. Finalmente, para aplicações que exigem uma elevada largura de banda, precisamos alocar largura de banda suficiente para ter certeza de que os pacotes não serão perdidos.

Tabela 8.9 Sensibilidade de aplicações a características do fluxo.

Aplicação	Confiabilidade	Atraso	*Jitter*	Largura de banda
FTP	Alta	Baixa	Baixa	Média
HTTP	Alta	Média	Baixa	Média
Áudio sob demanda	Baixa	Baixa	Alta	Média
Vídeo sob demanda	Baixa	Baixa	Alta	Alta
Voz sobre IP	Baixa	Alta	Alta	Baixa
Vídeo sobre IP	Baixa	Alta	Alta	Alta

8.5.2 Classes de fluxo

Com base nas características dos fluxos, podemos classificar os fluxos em grupos, de modo que cada grupo apresente o nível exigido de cada característica. A comunidade Internet ainda não

definiu tal classificação formalmente. No entanto, sabemos, por exemplo, que um protocolo como o FTP requer um alto nível de confiabilidade e, provavelmente, um nível médio de largura de banda; por outro lado, o nível de atraso e de *jitter* não é importante para esse protocolo.

Exemplo 8.11

Embora a Internet não tenha definido formalmente as classes de fluxo, o protocolo ATM o fez. De acordo com as especificações do ATM, existem cinco classes de serviço definidas.

a. **Taxa Constante de *Bits*** (CBR – Constant Bit Rate). Essa classe é usada para emular a comutação de circuitos. Aplicações do tipo CBR são muito sensíveis a variações no atraso das células. Exemplos de tráfegos CBR são o tráfego de telefonia, de videoconferências e de televisão.

b. **Taxa Variável de *Bits*-Não Tempo Real** (VBR-NRT – Variable Bit Rate-Non Real Time). Usuários nessa classe podem enviar tráfego a uma taxa que varia com o tempo dependendo da disponibilidade de dados do usuário. Um exemplo é o uso de multimídia em um e-mail.

c. **Taxa de *Bits* Variável-Tempo Real** (VBR-RT – Variable Bit Rate-Real Time). Essa classe é semelhante à VBR-NRT, mas é projetada para aplicações, como vídeo interativo comprimido que são sensíveis à variação de atraso das células.

d. **Taxa de *Bit* Disponível** (ABR – Available Bit Rate). Essa classe de serviços ATM fornece um controle de fluxo baseado na taxa de transferência e é voltada para tráfegos de dados, como transferência de arquivos e e-mail.

e. **Taxa de *Bits* Não Especificada** (UBR – Unspecified Bit Rate). Essa categoria inclui todas as outras classes e é amplamente utilizada atualmente para o TCP/IP.

8.5.3 Controle de fluxo para melhorar a QoS

Embora não haja na Internet uma definição formal das classes de fluxo, um datagrama IP tem um campo de Tipo de Serviço (ToS – Type of Service) que pode informalmente definir o tipo de serviço necessário por um conjunto de datagramas enviados por uma aplicação. Se atribuirmos a certo tipo de aplicação um único nível de serviço exigido, podemos, então, alocar alguns recursos para esses níveis de serviço. Isto pode ser feito usando vários mecanismos.

Escalonamento

O tratamento de pacotes (datagramas) na Internet com base no nível de serviço exigido por eles pode acontecer principalmente nos roteadores. É em um roteador que um pacote pode ser atrasado, sofrer *jitter*, ser perdido ou receber a largura de banda necessária. Uma boa técnica de escalonamento trata os diferentes fluxos de uma maneira justa e adequada. Diversas técnicas de escalonamento são projetadas para melhorar a qualidade de serviço. Aqui, discutimos três delas: *fila FIFO*, *fila de prioridade* e *fila justa ponderada*.

Fila FIFO

Nas **filas FIFO** (*First-In, First-Out*, ou Primeiro a Entrar, Primeiro a Sair), os pacotes esperam em um *buffer* (fila) até que o nó (roteador) esteja pronto para processá-los. Se a taxa média de chegada for maior do que a taxa média de processamento, a fila encherá e novos pacotes serão descartados. Uma fila FIFO é algo familiar para aqueles que já tiveram que esperar por um ônibus em uma parada de ônibus. A Figura 8.59 mostra uma visão conceitual de uma fila FIFO. E também a relação temporal entre a chegada e saída de pacotes nessa fila. Pacotes provenientes de diferentes aplicações (com diferentes tamanhos) chegam à fila, são processados e saem. Um pacote maior pode definitivamente precisar de um tempo de processamento maior. Na figura, os

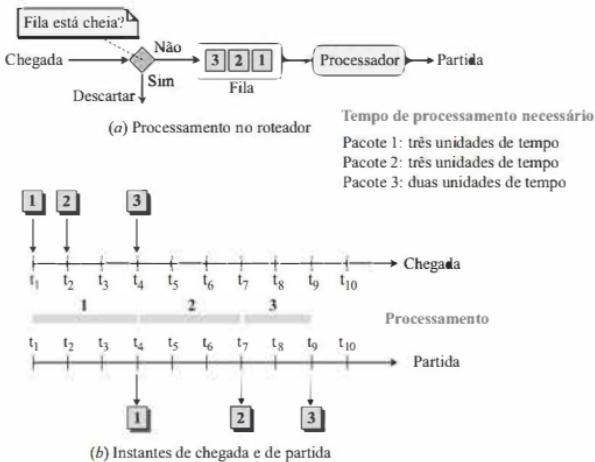
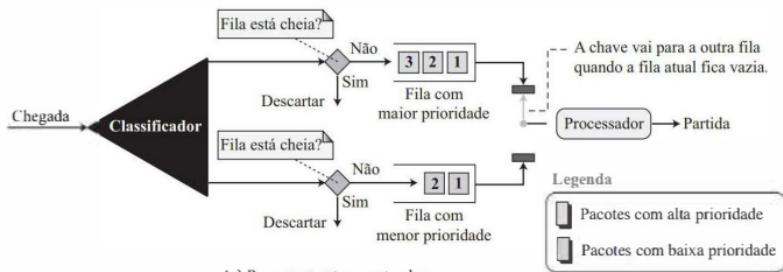
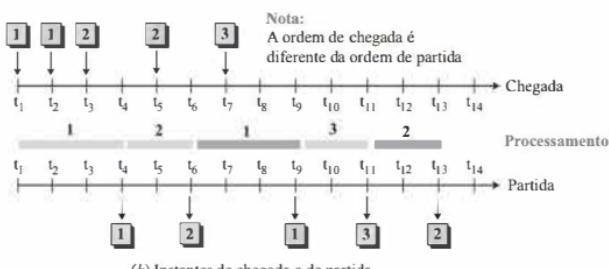



Figura 8.59 Fila FIFO.

pacotes 1 e 2 precisam de três unidades de tempo de processamento, mas o pacote 3, que é menor, requer duas unidades de tempo. Isto significa que os pacotes podem chegar com alguns atrasos, mas partir com atrasos diferentes. Se os pacotes pertencem à mesma aplicação, isto cria *jitters*. Se os pacotes pertencem a diferentes aplicações, isto também gera *jitters* para cada aplicação.

O escalonamento usando filas FIFO é o padrão na Internet. A única coisa garantida nesse tipo de fila é que os pacotes saem na mesma ordem em que chegam. As filas FIFO fazem alguma distinção entre classes de pacotes? A resposta é definitivamente não. O escalonamento com esse tipo de fila é o que vemos na Internet, sem diferenciação de serviços entre os pacotes de diferentes fontes. Com filas FIFO, todos os pacotes são tratados da mesma forma em uma rede de comutação. Independentemente de o pacote pertencer ao FTP, a um aplicativo de Voz sobre IP ou ser uma mensagem de e-mail, todos eles ficarão igualmente sujeitos a perdas, *jitter* e atraso. A largura de banda alocada para cada aplicação depende de quantos pacotes chegam ao roteador em um intervalo de tempo. Se precisarmos fornecer serviços distintos para diferentes classes de pacotes, precisamos ter outros mecanismos de escalonamento.

Fila de prioridade

O atraso de fila em filas FIFO muitas vezes degrada a qualidade do serviço na rede. Um quadro transportando pacotes em tempo real pode ter que esperar por um longo período atrás de um quadro com um pequeno arquivo. Resolvemos esse problema usando múltiplas filas e filas de prioridade.

Nas **filas de prioridade**, os pacotes primeiramente recebem uma classe de prioridade. Cada classe de prioridade tem sua própria fila. Os pacotes na fila de maior prioridade são processados primeiro, e os na fila de menor prioridade são processados por último. Perceba que o sistema não deixa de servir uma fila até que ela esteja vazia. A prioridade de um pacote é determinada por um campo específico no cabeçalho dele: o campo de ToS em um cabeçalho IPv4, o campo de prioridade do IPv6, um número de prioridade atribuído a um endereço de destino, um número de prioridade atribuído a uma aplicação (número da porta de destino) e assim por diante.

A Figura 8.60 mostra uma fila de prioridade com dois níveis de prioridade (para simplificar). Uma fila de prioridade pode prover uma melhor QoS do que a fila FIFO, porque o tráfego de maior prioridade, como o tráfego multimídia, pode chegar ao destino com um menor atraso. No entanto, existe uma desvantagem em potencial. Se houver um fluxo contínuo em uma fila de alta prioridade, os pacotes nas filas de menor prioridade nunca terão uma chance de serem processados. É uma condição conhecida como inanição (*starvation*). Uma inanição grave pode resultar no descarte de alguns pacotes de prioridade mais baixa. Na figura, os pacotes de maior prioridade são enviados antes dos pacotes de prioridade mais baixa.

(*a*) Processamento no roteador

(*b*) Instantes de chegada e de partida

Figura 8.60 Fila de prioridade.

Fila justa ponderada

Um método melhor de escalonamento consiste no uso de ***filas justas ponderadas***. Nessa técnica, ainda são atribuídas diferentes classes aos pacotes e eles ainda são colocados em filas diferentes. As filas, entretanto, são ponderadas com base na prioridade; uma maior prioridade significa um peso maior. O sistema processa pacotes de dados em cada fila usando um escalonamento do tipo *round-robin**, no qual o número de pacotes selecionados de cada fila é baseado no peso correspondente dela. Por exemplo, se os pesos forem 3, 2 e 1, são processados três pacotes da primeira fila, dois da segun-

* N. de T.: O *round-robin* é um algoritmo de escalonamento no qual todas as filas são visitadas de forma circular, gastando uma unidade de tempo (ou algumas, caso cada fila tenha um peso distinto) no processamento de cada uma delas.

da e um da terceira. Dessa forma, temos uma fila justa com prioridades. A Figura 8.61 mostra a técnica com três classes de prioridade. Em filas justas ponderadas, cada classe pode receber uma pequena quantidade de tempo em cada intervalo de tempo. Em outras palavras, uma fração do tempo é dedicada a servir cada classe de pacotes, mas a fração depende da prioridade da classe.

Por exemplo, na figura, se a vazão do roteador for R, a classe com a maior prioridade pode ter uma vazão do R/2, a classe com prioridade média pode ter uma vazão de R/3 e a classe com a prioridade mais baixa pode ter uma vazão de R/6. No entanto, essa situação é verdadeira se todas as três classes tiverem o mesmo tamanho de pacote, o que pode não acontecer. Pacotes de tamanhos diferentes podem criar muitos desequilíbrios ao se tentar distribuir uma parcela razoável de tempo entre as diferentes classes.

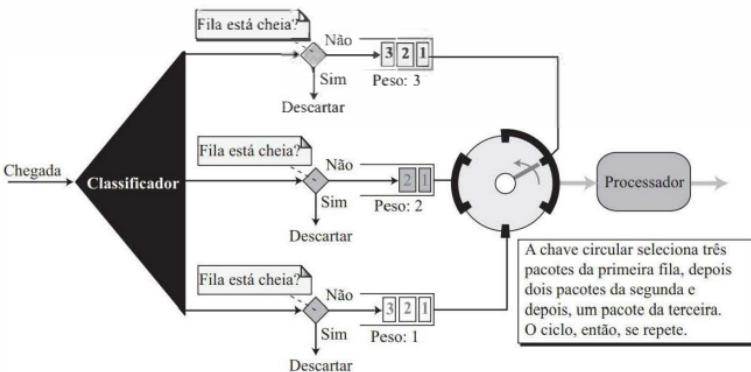



Figura 8.61 Fila justa ponderada.

Conformação ou policiamento de tráfego

O ato de controlar a quantidade e a taxa de tráfego é denominado conformação de tráfego (do inglês, *traffic shaping*) ou policiamento de tráfego. O primeiro termo é usado quando o tráfego deixa uma rede; o segundo, quando os dados entram na rede. Duas técnicas podem conformar ou policiar o tráfego: o balde furado (*leaky bucket*) e o balde de fichas (*token bucket*).

Balde furado

Se um balde tem um pequeno furo na parte inferior, a água vaza dele a uma taxa constante enquanto houver água ali. A taxa em que a água vaza não depende da taxa de entrada dela, a não ser que o balde esteja vazio. Se o balde estiver cheio, a água transborda. A taxa de entrada pode variar, mas a taxa de saída permanece constante. De forma similar, no contexto de redes, uma técnica chamada **balde furado** (*leaky bucket*) pode suavizar o tráfego em rajadas. Blocos enviados em rajadas são armazenados no balde e transmitidos a uma taxa média. A Figura 8.62 mostra um balde furado e seus efeitos.

Na Figura, consideremos que a rede tenha se comprometido a prover uma largura de banda de 3 Mbps para uma estação. A utilização do balde furado molda o tráfego de entrada para que ele fique em conformidade com esse compromisso. Na Figura 8.62, a estação envia uma rajada de dados a uma velocidade de 12 Mbps durante dois segundos, resultando em um total de 24 Mb de dados. A estação fica em silêncio por cinco segundos e em seguida envia dados a uma taxa de 2 Mbps durante três segundos, resultando em um total de 6 Mb de dados. Ao todo, a estação enviou 30 Mb de dados em 10 se-

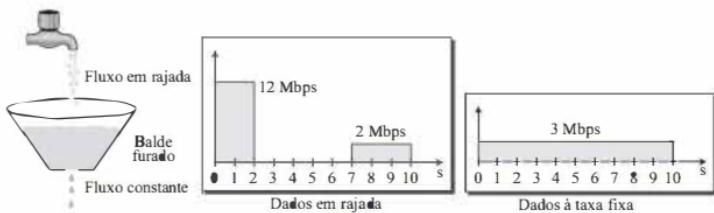



Figura 8.62 Balde furado (*leaky bucket*).

gundos. O balde furado suaviza o tráfego, enviando dados a uma taxa de 3 Mbps durante os mesmos 10 segundos. Sem o balde furado, a rajada inicial poderia prejudicar a rede por consumir mais banda do que o valor reservado para essa estação. Também podemos ver que o balde furado pode ajudar a evitar congestionamentos.

Uma implementação simples do balde furado é mostrada na Figura 8.63. Uma fila FIFO armazena os pacotes. Se o tráfego for constituído de pacotes de tamanho fixo (por exemplo, células em redes ATM), o processo remove um número fixo de pacotes da fila a cada pulso do relógio. Se o tráfego for constituído por pacotes de tamanho variável, a taxa de saída fixa deve se basear no número de *bytes* ou *bits*.

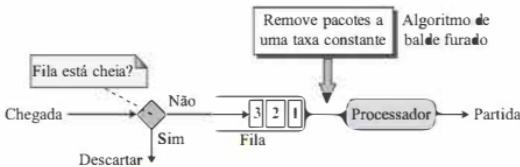



Figura 8.63 Implementação do balde furado.

A seguir, é apresentado um algoritmo para pacotes de comprimento variável:

1. Inicializar um contador em n no pulso do relógio.
2. Se n for maior que o tamanho do pacote, enviar o pacote e decrementar o contador de uma quantidade igual ao tamanho do pacote. Repetir esse passo até que o valor do contador seja menor que o tamanho do pacote.
3. Reiniciar o contador em n e ir para o passo 1.

Um algoritmo de balde furado transforma o tráfego em rajada em um tráfego de taxa fixa com a taxa de transferência média dos dados. Ele pode descartar os pacotes se o balde estiver cheio.

Balde de fichas

O balde furado é muito restritivo. Uma estação inativa não acumula créditos. Por exemplo, se uma estação não enviar dados por algum tempo, seu balde fica vazio. Agora, se a estação tiver

dados em rajada para enviar, o balde furado permite apenas uma taxa média de envio. O período de tempo em que a estação permaneceu inativa não é levado em consideração. Por outro lado, o algoritmo de **balde de fichas** (*token bucket*) permite que estações ociosas acumulem crédito para o futuro na forma de fichas.

Considere que a capacidade do balde seja de c fichas e que as fichas entram no balde a uma taxa de r fichas por segundo. O sistema remove uma ficha para cada célula de dados enviada. O número máximo de células que podem entrar na rede durante qualquer intervalo de comprimento t é mostrado a seguir.

Número máximo de pacotes = $rt + c$

A taxa de transferência média máxima para o balde de fichas é mostrada a seguir.

Taxa média máxima = $(rt + c)/t$ pacotes por segundo

Isto significa que o balde de fichas limita a taxa média de pacotes enviados para a rede. A Figura 8.64 ilustra a ideia.

Figura 8.64 Balde de fichas.

Exemplo 8.12

Consideremos que a capacidade do balde seja de 10.000 fichas e que as fichas são adicionadas a uma taxa de r fichas por segundo. Se o sistema ficar inativo durante 10 segundos (ou mais), o balde coleta 10.000 fichas e fica cheio. Quaisquer fichas adicionais serão descartadas. A taxa média máxima é mostrada a seguir.

Taxa média máxima = $(1.000t + 10.000)/t$

O balde de fichas pode ser facilmente implementado com um contador, que é inicializado em zero. Cada vez que uma ficha é adicionada, o contador é incrementado de 1. Cada vez que uma unidade de dados é enviada, o contador é decrementado de 1. Quando o contador atinge zero, a estação não pode enviar dados.

O balde de fichas permite tráfego em rajada a uma taxa máxima controlada.

Combinando o balde de fichas e o balde furado

As duas técnicas podem ser combinadas para permitir que uma estação inativa acumule crédito e, ao mesmo tempo, para regular o tráfego. O balde furado é aplicado após o balde de fichas; a taxa do balde furado precisa ser superior à taxa de fichas que entram no balde.

Reserva de recursos

Um fluxo de dados requer recursos, como um *buffer*, largura de banda, tempo de CPU e assim por diante. A qualidade do serviço é melhorada se esses recursos forem reservados antecipadamente. A seguir, discutimos um modelo de QoS denominado Serviços Integrados, que depende fortemente da reserva de recursos para melhorar a qualidade de serviço.

Controle de admissão

O controle de admissão refere-se ao mecanismo usado por um roteador ou por um *switch* para aceitar ou rejeitar um fluxo com base em parâmetros predefinidos denominados *especificações de fluxo*. Antes de um roteador aceitar um fluxo para processamento, ele verifica as especificações do fluxo para ver se sua capacidade é suficiente para lidar com o novo fluxo. Ele leva em conta a largura de banda, o tamanho do *buffer*, a velocidade da CPU, etc., bem como os seus comprometimentos anteriores com outros fluxos. O controle de admissão em redes ATM é conhecido como Controle de Admissão de Conexões (CAC – Connection Admission Control), uma parte importante da estratégia para controlar congestionamentos.

8.5.4 Serviços Integrados

A Internet tradicional fornecia apenas o serviço de entrega de melhor esforço a todos os usuários, independentemente das suas necessidades. Algumas aplicações, no entanto, precisam de uma quantidade mínima de largura de banda para funcionar (por exemplo, áudio e vídeo em tempo real). Para fornecer diferentes níveis de QoS para aplicações distintas, a IETF (discutida no Capítulo 1) desenvolveu o modelo de **Serviços Integrados** (IntServ – Integrated Services). Nele, que é uma arquitetura *baseada em fluxo*, recursos como a largura de banda são explicitamente reservados para um determinado fluxo de dados. Em outras palavras, o modelo considera um requisito específico de uma aplicação em um caso particular, independentemente do tipo de aplicação (transferência de dados, voz sobre IP ou vídeo sob demanda). O importante são os recursos que a aplicação precisa, não o que ela está fazendo.

O modelo é baseado em três esquemas:

1. Os pacotes são primeiramente classificados de acordo com o serviço de que eles precisam.
2. O modelo utiliza escalonamento para encaminhar os pacotes de acordo com as características de seus fluxos.
3. Dispositivos como roteadores usam *controle de admissão* para determinar se aquele dispositivo tem a capacidade (recursos disponíveis para tratar o fluxo) de atender o fluxo antes de se comprometer a fazê-lo. Por exemplo, se um aplicativo exige uma elevada taxa de transferência de dados, mas um roteador no caminho não é capaz de fornecê-la, ele recusa a admissão.

Antes de discutirmos esse modelo, precisamos enfatizar que ele é baseado em fluxos, o que significa que todas as reservas têm de ser feitas antes que um fluxo possa começar. Isto implica que precisamos de um serviço orientado à conexão na camada de rede. É preciso haver uma fase de estabelecimento de conexão para informar todos os roteadores dos requisitos e para obter suas aprovações (controle de admissão). Entretanto, como o IP é, na realidade, um protocolo não orientado à conexão, precisamos executar outro protocolo sobre o IP para transformá-lo em

um protocolo orientado à conexão antes que possamos usar esse modelo. Esse protocolo é denominado **Protocolo de Reserva de Recursos** (RSVP – Resource Reservation Protocol) e será discutido mais adiante.

> **Serviços Integrados é um modelo de QoS baseado em fluxos projetado para o IP. Nesse modelo, os pacotes são marcados por roteadores de acordo com as caracteristicas do fluxo.**

Especificação do fluxo

Dissemos que o IntServ é baseado em fluxos. Para definir um fluxo específico, uma fonte precisa definir uma *especificação de fluxo*, composta por duas partes:

1. **Rspec (especificação de recursos).** O Rspec define o recurso que o fluxo precisa reservar (*buffer*, largura de banda, etc.).
2. **TSpec (especificação de tráfego).** O TSpec define a caracterização de tráfego do fluxo, algo que discutimos anteriormente.

Admissão

Após um roteador ter recebido a especificação de fluxo de uma aplicação, ele decide aceitar ou recusar o serviço. A decisão baseia-se nos comprometimentos anteriores do roteador e na sua disponibilidade atual de recursos.

Classes de serviço

Duas classes de serviços foram definidas para os Serviços Integrados: serviço garantido e serviço de carga controlada.

Classe de serviço garantido

Esse tipo de serviço foi concebido para tráfego em tempo real que precisa de garantias de um reduzido atraso fim a fim. O atraso fim a fim corresponde à soma dos atrasos nos roteadores, do atraso de propagação no meio e dos mecanismos de configuração. Apenas o primeiro, a soma dos atrasos nos roteadores, pode ser garantido pelo roteador. Esse tipo de serviço garante que os pacotes chegarão dentro de um determinado tempo de entrega e não serão descartados se o tráfego do fluxo permanecer dentro do limite informado no *TSpec*. Podemos dizer que serviços garantidos são *serviços quantitativos*, nos quais a quantidade de atraso fim a fim e a taxa de transferência de dados devem ser definidas pela aplicação. Normalmente, serviços garantidos são necessários para aplicações em tempo real (voz sobre IP).

Classe de serviço de carga controlada

Esse tipo de serviço foi projetado para aplicações que podem aceitar alguns atrasos, mas que são sensíveis a uma rede sobrecarregada e ao perigo de perder pacotes. Bons exemplos desse tipo de aplicação são transferência de arquivos, e-mail e acesso à Internet. O serviço de carga controlada é um *serviço qualitativo*, no qual o aplicativo solicita a possibilidade de ter um serviço com reduzida ou nenhuma perda de pacotes.

Protocolo de Reserva de Recursos

Dissemos que o modelo de Serviços Integrados requer uma camada de rede orientada à conexão. Como o IP é um protocolo não orientado à conexão, um novo protocolo foi projetado para operar

em cima do IP de modo a torná-lo orientado à conexão. Um protocolo orientado à conexão precisa apresentar as fases de estabelecimento e de encerramento de conexão, conforme discutimos no Capítulo 4. Antes de discutirmos o Protocolo de Reserva de Recursos (RSVP), é preciso mencionar que ele é um protocolo independente, separado do modelo de Serviços Integrados. Pode ser utilizado em outros modelos no futuro.

Árvores *multicast*

O RSVP, no entanto, é diferente de outros protocolos orientados à conexão no sentido que ele se baseia em comunicações *multicast*. Entretanto, o RSVP também pode ser usado para comunicação *unicast*, já que o *unicast* é simplesmente um caso especial de *multicast* no qual há apenas um membro no grupo *multicast*. A razão para essa abordagem é permitir que o RSVP forneça a reserva de recursos para todos os tipos de tráfego, incluindo multimídia, que muitas vezes utiliza comunicações *multicast*.

Reserva baseada no receptor

No RSVP, os destinatários, e não o remetente, fazem a reserva. Essa estratégia coincide com a de outros protocolos baseados em *multicast*. Por exemplo, em protocolos de roteamento *multicast*, os destinatários, e não o remetente, tomam a decisão de entrar ou sair de um grupo *multicast*.

Mensagens RSVP

O RSVP tem diversos tipos de mensagens. Entretanto, para nossos propósitos, discutimos apenas dois deles: *Caminho* e *Resv*.

- **Mensagens de caminho.** Lembre-se de que os destinatários de um fluxo fazem a reserva no RSVP. No entanto, eles não sabem o caminho percorrido pelos pacotes antes que a reserva seja efetuada. O caminho é necessário para a reserva. Para resolver o problema, o RSVP utiliza mensagens de *Caminho*. Uma mensagem de *Caminho* sai do remetente e atinge todos os destinatários no caminho *multicast*. Durante o trajeto, uma mensagem de Caminho armazena as informações necessárias para os destinatários. Ela é enviada em um ambiente *multicast*; uma nova mensagem é criada quando o caminho diverge. A Figura 8.65 mostra as mensagens de Caminho.

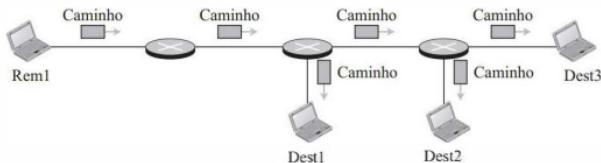



Figura 8.65 Mensagens de caminho.

- **Mensagens de Resv.** Após um receptor ter recebido uma mensagem de Caminho, ele envia uma mensagem de reserva, ou *Resv*. A mensagem de *Resv* viaja para o remetente (na direção contrária ao fluxo) e faz uma reserva de recursos nos roteadores com suporte ao RSVP. Se um roteador no caminho não tem suporte ao RSVP, ele encaminha o pacote com base nos métodos de entrega de melhor esforço que discutimos anteriormente. A Figura 8.66 mostra as mensagens de Resv.

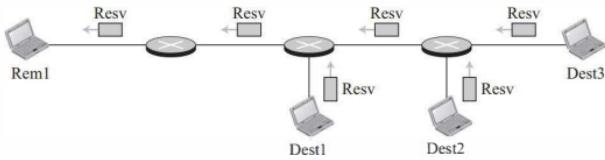



Figura 8.66 Mensagens de reserva (Resv).

Fusão de reservas No RSVP, os recursos não são reservados para cada destinatário em um fluxo; as reservas são fundidas. Na Figura 8.67, a estação Dest3 solicita uma largura de banda de 2 Mbps, enquanto Dest2 solicita uma largura de banda de 1 Mbps. O roteador R3, que precisa fazer uma reserva de banda, funde as duas solicitações. A reserva é feita para 2 Mbps, o maior dos dois valores, porque uma reserva de entrada de 2 Mbps é capaz de lidar com ambas as solicitações. A mesma situação ocorre para R2. O leitor pode se perguntar por que Dest2 e Dest3, que pertencem ambos a um mesmo fluxo, solicitariam diferentes quantidades de largura de banda. A resposta é que, em um ambiente multimídia, destinatários diferentes podem lidar com diferentes níveis de qualidade. Por exemplo, Dest2 pode ser capaz de receber vídeo a apenas 1 Mbps (menor qualidade), enquanto Dest3 pode querer receber vídeo a 2 Mbps (maior qualidade).

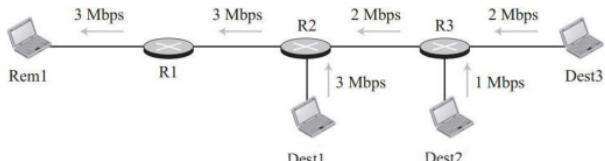



Figura 8.67 Fusão de reservas.

Estilos de reserva Quando há mais do que um fluxo, o roteador precisa fazer uma reserva para acomodar todos eles. O RSVP define três tipos de estilos de reserva: *filtro com curingas* (FC), *filtro fixo* (FF) e *filtro explícito compartilhado* (EC).

- **Estilo de filtro com curingas.** Nesse estilo, também conhecido como *Filtro com Wild Card*, o roteador cria uma única reserva para todos os emissores. A reserva baseia-se na maior solicitação. Esse tipo de estilo é utilizado quando os fluxos de emissores diferentes não ocorrem ao mesmo tempo.
- **Estilo de filtro fixo.** Nesse estilo, o roteador cria uma reserva distinta para cada fluxo. Isto significa que se houver *n* fluxos, *n* reservas diferentes são feitas. Esse tipo de estilo é usado quando há uma alta probabilidade de que fluxos de diferentes emissores ocorram ao mesmo tempo.
- **Estilo de explícito compartilhado.** Nesse estilo, o roteador cria uma única reserva que pode ser compartilhada por um conjunto de fluxos.

Soft state As informações de reserva (estado) armazenadas em cada nó para um fluxo precisam ser atualizadas periodicamente. Isto é conhecido como *soft state*, ou *estado flexível*, em comparação com o *hard state* (*estado rígido*) utilizado em outros protocolos de circuitos virtuais, como o ATM, nos quais a informação sobre o fluxo é mantida até que ela seja apagada. O intervalo-padrão de atualização (reserva do *soft state*) é atualmente de 30 segundos.

Problemas com o modelo de Serviços Integrados

Há pelo menos dois problemas com o modelo de Serviços Integrados que podem impedir a sua aplicação integral na Internet: escalabilidade e limitação do tipo de serviço.

Escalabilidade

O modelo de Serviços Integrados exige que cada roteador mantenha as informações para cada fluxo. Como a Internet está crescendo a cada dia, esse é um problema sério. Manter informações é especialmente problemático para roteadores no núcleo da rede, pois eles são projetados principalmente para comutar pacotes em alta velocidade, não para processar informações.

Limitação do tipo de serviço

O modelo de Serviços Integrados oferece apenas dois tipos de serviços, garantidos e de carga controlada. Aqueles que se opõem a esse modelo argumentam que as aplicações podem precisar de mais do que esses dois tipos de serviços.

8.5.5 Serviços Diferenciados

Nesse modelo, também conhecido como **DiffServ** (*Differentiated Services*, ou Serviços Diferenciados), os pacotes são marcados pelas aplicações como pertencentes a classes de acordo com as suas prioridades. Roteadores e *switches*, usando diversas estratégias de filas, roteiam os pacotes. Esse modelo foi introduzido pela Força-Tarefa de Engenharia da Internet (IETF – Internet Engineering Task Force) para lidar com as deficiências do modelo de Serviços Integrados (IntServ). Duas alterações fundamentais foram feitas:

1. O processamento principal foi movido do núcleo da rede para a borda da rede. Isto resolve o problema de escalabilidade. Os roteadores não têm que armazenar informações sobre os fluxos. As aplicações, ou as estações, definem o tipo de serviço de que precisam toda vez que enviam um pacote.
2. O serviço por fluxo é transformado em um serviço por classe. O roteador roteia o pacote com base na classe de serviço definida no pacote, não no fluxo. Isto resolve o problema da limitação do tipo de serviço. Podemos definir diferentes tipos de classes com base nas necessidades das aplicações.

Serviços Diferenciados é um modelo de QoS baseado em classes projetado para o IP. Nesse modelo, os pacotes são marcados pelas aplicações de acordo com sua prioridade.

Campo DS

No DiffServ, cada pacote tem um campo denominado campo DS. O valor desse campo é definido na fronteira da rede, pela estação ou pelo primeiro roteador designado como roteador de borda. A IETF propõe a substituição do campo de Tipo de Serviço (ToS – Type of Service) existente no IPv4 ou a classe de prioridade no IPv6 pelo campo de DS, conforme mostrado na Figura 8.68.

Figura 8.68 Campo DS.

O campo DS contém dois subcampos: DSCP e CU. O Código de Serviços Diferenciados (DSCP – Differentiated Services Code Point) é um subcampo de 6 *bits* que define o **comportamento por salto** (PHB – Per-Hop Behavior). O subcampo Não Utilizado Atualmente (CU – Currently Unused), de 2 *bits*, não é usado atualmente.

Um nó (roteador) com suporte ao DiffServ utiliza os 6 *bits* do DSCP como um índice para uma tabela que define o mecanismo de tratamento para o pacote sendo processado no momento.

Comportamento por salto

O modelo DiffServ define comportamentos por salto (PHBs – Per-Hop Behaviors) para cada nó que recebe um pacote. Até agora, três PHBs foram definidos: DE-PHB, EF-PHB e AF-PHB.

- **DE-PHB.** O DE-PHB (*Default PHB*, ou PHB padrão) é equivalente ao serviço de entrega de melhor esforço, o que é compatível com o campo de ToS.
- **EF-PHB.** O EF-PHB (*Expedited Forwarding PHB*, ou PHB de Encaminhamento Expresso) fornece os seguintes serviços:
  a. Baixa perda.
  b. Baixa latência.
  c. Garantia de largura de banda.

Isto é equivalente a ter uma conexão virtual entre a origem e o destino.

- **AF-PHB.** O AF-PHB (*Assured Forwarding PHB*, ou PHB de Encaminhamento Assegurado) entrega o pacote com uma alta confiabilidade, contanto que a classe de tráfego não exceda o perfil de tráfego do nó. Os usuários da rede devem estar cientes de que alguns pacotes podem ser descartados.

Condicionamento de tráfego

Para implementar o modelo DiffServ, o nó DS usa condicionadores de tráfego, como medidores, marcadores, conformadores e descartadores, conforme mostra a Figura 8.69.

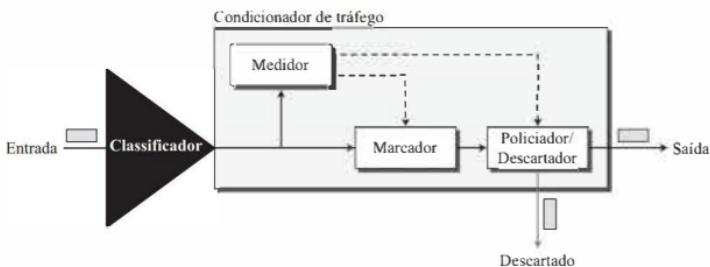



Figura 8.69 Condicionamento de tráfego.

- **Medidores.** O medidor verifica se o fluxo de entrada corresponde ao perfil de tráfego negociado. O medidor também envia esse resultado a outros componentes. O medidor pode usar diversas ferramentas, como um balde de fichas, para verificar o perfil.

- **Marcador.** Um marcador pode remarcar um pacote que está usando entrega de melhor esforço (DSCP: 000000) ou rebaixar a marcação de um pacote com base em informações recebidas do medidor. O processo de rebaixar a marca (baixando a classe do fluxo) ocorre se o fluxo não coincidir com o perfil. Um marcador não eleva a marca de um pacote (não o promove de classe).
- **Policiador.** Um policiador (*shaper*) utiliza as informações recebidas do medidor para remodelar o tráfego, caso ele não esteja compatível com o perfil negociado.
- **Descartador.** Um descartador (*dropper*), que funciona como um policiador sem *buffer*, descarta pacotes caso o fluxo viole severamente o perfil negociado.

8.6 MATERIAL DO FINAL DO CAPÍTULO

8.6.1 Leitura recomendada

Para mais detalhes sobre os assuntos discutidos neste capítulo, recomendamos os seguintes livros e RFCs. Os itens entre colchetes [...] referem-se à lista de referências no final do texto.

Livros

Diversos livros cobrem alguns aspectos de multimídia: [Com 06], [Tan 03] e [G & W 04].

RFCs

Diversas RFCs mostram diferentes atualização de tópicos discutidos neste capítulo, incluindo RFC 2198, RFC 2250, RFC 2326, RFC 2475, RFC 3246, RFC 3550 e RFC 3551.

8.6.2 Termos-chave

- algoritmo do balde furado (*leaky bucket*)
- associação
- Áudio MPEG Camada 3 (MP3 – MPEG Audio Layer 3)
- balde de fichas
- *buffer* de reprodução
- carimbo de tempo (*timestamp*)
- codificação aritmética
- codificação de Huffman
- codificação perceptual
- codificação preditiva
- Codificação Preditiva Linear (LPC – Linear Predictive Coding)
- codificação RLE (*run-length encoding*)
- comportamento por salto (PHB – Per-Hop Behavior)
- compressão com perdas
- compressão espacial
- compressão sem perdas
- compressão temporal
- Correção Antecipada de Erros (FEC – Forward Error Correction)
- DM Adaptativa (ADM – Adaptive DM)
- DPCM Adaptativa (ADPCM – Adaptive DPCM)
- fila de prioridade
- fila FIFO (*FirstIn, First-Out*, ou Primeiro a Entrar, Primeiro a Sair)
- fila justa ponderada
- gatekeeper
- *gateway*
- Grupo de Especialistas em Fotografia (JPEG – Joint Photographic Experts Group)

- Grupo de Especialistas em Imagens em Movimento (MPEG – Motion Picture Experts Group)
- H.323
- *jitter*
- Lempel-Ziv-Welch (LZW)
- mascaramento em frequência
- mascaramento temporal
- metarquivo
- *mixer*
- Modulação Delta (DM – Delta Modulation)
- PCM Diferencial (DPCM – Differential PCM)
- Protocolo de Controle de Fluxo de Transmissão (SCTP – Stream Control Transmission Protocol)
- Protocolo de Controle de Transporte em Tempo Real (RTCP – Real-time Transport Control Protocol)
- Protocolo de Fluxo Contínuo em Tempo Real (RTSP – Real-Time Streaming Protocol)
- Protocolo de Iniciação de Sessão (SIP – Session Initiation Protocol)
- Protocolo de Reserva de Recursos (RSVP – Resource Reservation Protocol)
- Protocolo de Transporte em Tempo Real (RTP – Real-time Transport Protocol)
- psicoacústica
- quadro bidirecional (quadro B)
- quadro intracodificado (quadro I)
- quadro predito (quadro P)
- Qualidade de Serviço (QoS – Quality of Service)
- serviço de múltiplos fluxos
- serviço multiprovedor (*multihoming*)
- Serviços Diferenciados (DS ou DiffServ – Differentiated Services)
- Serviços Integrados (IntServ – Integrated Services)
- servidor de mídia
- servidor de registro
- Transformada Discreta do Cosseno (DCT – Discrete Cosine Transform)
- Voz sobre IP (VoIP – Voice over IP)

8.6.3 Resumo

Podemos dividir a compressão em duas grandes categorias: compressão sem perdas e com perdas. Na compressão sem perdas, a integridade dos dados é preservada, pois os algoritmos de compressão e descompressão são inversos exatos um do outro: nenhuma parte dos dados é perdida no processo. A compressão com perdas não é capaz de preservar a precisão dos dados, mas tem a vantagem de reduzir o tamanho dos dados comprimidos.

Arquivos de áudio/vídeo podem ser obtidos da Internet para uso futuro (fluxo contínuo de áudio/vídeo armazenado) ou transmitido aos clientes por meio da Internet (fluxo contínuo de áudio/vídeo ao vivo). A Internet também pode ser usada para áudio/vídeo interativo ao vivo. Áudio e vídeo precisam ser digitalizados antes de serem enviados através da Internet. Podemos usar um servidor Web, um servidor Web com um metarquivo, um servidor de mídia, ou um servidor de mídia e o RTSP para obter um arquivo de fluxo contínuo (*streaming*) de áudio/vídeo.

Dados em tempo real em uma rede de comutação de pacotes exigem que seja preservada a relação temporal entre os pacotes de uma sessão. Lacunas entre pacotes consecutivos no receptor causam um fenômeno conhecido como *jitter*, que pode ser controlado usando carimbos de tempo (*timestamps*) e por meio de uma escolha adequada do tempo de reprodução.

Voz sobre IP é uma aplicação de áudio/vídeo interativo em tempo real. O Protocolo de Iniciação de Sessão (SIP – Session Initiation Protocol) é um protocolo de camada de aplicação que estabelece, gerencia e encerra sessões multimídia. O H.323 é um padrão ITU que permite que um telefone conectado a uma rede de telefonia pública consiga se comunicar com um computador conectado à Internet.

O tráfego multimídia em tempo real requer o uso tanto do UDP como do Protocolo de Transporte em Tempo Real (RTP – Real-Time Transport Protocol). O RTP trata os carimbos de tempo, sequenciamento e mixagem. O Protocolo de Controle de Transporte em Tempo Real (RTCP – Real-Time Transport Control Protocol) fornece controle de fluxo, controle da qualidade dos dados e informações de retorno para as fontes de dados multimídia. O novo protocolo da camada de transporte, o SCTP, foi projetado para lidar com aplicações multimídia, como Voz sobre IP.

Escalonamento, conformação (ou policiamento) de tráfego, reserva de recursos e controle de admissão são técnicas usadas para melhorar a qualidade de serviço (QoS). Filas FIFO, filas de prioridade e filas justas ponderadas são técnicas de escalonamento. O balde furado (*leaky bucket*) e o balde de fichas (*token bucket*) são técnicas de conformação de tráfego. Serviços Integrados (IntServ – Integrated Services) é um modelo de QoS baseado em fluxo que foi projetado para o IP. O Protocolo de Reserva de Recursos (RSVP – Resource Reservation Protocol) é um protocolo de sinalização que ajuda o IP a criar um fluxo e a reservar recursos. Serviços Diferenciados (DiffServ – Differentiated Services) é um modelo de QoS baseado em classes que foi projetado para o IP.

8.7 ATIVIDADES PRÁTICAS

8.7.1 Testes

Diversos testes interativos relativos a este capítulo podem ser encontrados no *site* www.mhhe.com/forouzan. Recomendamos que o estudante realize os testes para verificar o seu entendimento do material apresentado antes de continuar com as atividades práticas.

Questões

8-1 Na codificação baseada em dicionário, se houver 60 caracteres na mensagem, quantas iterações do laço no algoritmo de compressão são executadas? Explique.

8-2 Na codificação baseada em dicionário, todas as entradas do dicionário criadas no processo devem ser usadas para a codificação ou para a decodificação?

8-3 Em um alfabeto com 20 símbolos, qual é o número de folhas em uma árvore de Huffman?

8-4 O código a seguir é um código instantâneo? Explique.

00 01 10 11 001 011 111

8-5 Considere que uma mensagem contém quatro tipos de caracteres (A, B, C e D) com a mesma probabilidade de ocorrência. Crie uma tabela de codificação de Huffman para essa mensagem. A codificação nesse caso chega a diminuir o número de *bits* a serem enviados?

8-6 Na codificação aritmética, duas mensagens diferentes podem ser codificadas no mesmo intervalo? Explique.

8-7 Na codificação preditiva, quais as diferenças entre DM e ADM?

8-8 Na codificação preditiva, quais as diferenças entre DPCM e ADPCM?

8-9 Compare o número de *bits* transmitidos para cada amostra PCM e DM se o valor máximo quantizado for

a. 12 **b.** 30 **c.** 50

8-10 Responda às seguintes perguntas sobre codificação preditiva:

a. O que significam na codificação DM: a distorção por excesso de inclinação (*slope overload distortion*) e distorção por ruído granular?

b. Explique como a codificação ADM resolve os erros anteriores.

8-11 Qual é a diferença entre o DM e o DPCM?

8-12 Qual é o problema em usar o método LPC para compactar sinais de voz?

8-13 Na codificação por transformada, quando um remetente envia a matriz M para um destinatário, o remetente precisa enviar a matriz T utilizada no cálculo? Explique.

8-14 No JPEG, precisamos de menos do que 24 *bits* para cada *pixel* se nossa imagem estiver usando uma ou duas cores primárias? Explique.

8-15 No JPEG, explique por que os valores de Q(m, n) na matriz de quantização não são idênticos. Em outras palavras, por que cada elemento em M(m, n) não é dividido por um valor fixo em vez de um valor diferente?

8-16 Explique por que usar a matriz Q10 resulta em um melhor índice de compressão e em uma menor qualidade da imagem do que usar a matriz Q90 (ver Figura 8.18).

8-17 No JPEG, explique por que precisamos arredondar o resultado da divisão na etapa de quantização.

8-18 Explique por que a combinação dos passos de quantização/dequantização no JPEG é um processo com perdas apesar de dividirmos cada elemento da matriz M pelo valor correspondente na matriz Q quando fazemos a quantificação e multiplicarmos os elementos pelos mesmos valores quando fazemos a dequantização.

8-19 Em uma comunicação multimídia, considere que um emissor só consegue codificar uma imagem utilizando a codificação JPEG, mas o receptor em potencial só consegue decodificar uma imagem se ela for codificada em GIF. Essas duas entidades são capazes de trocar dados multimídia?

8-20 Quando transmitimos áudio/vídeo armazenado na forma de um fluxo contínuo (*streaming*), qual é a diferença entre a primeira abordagem (Figura 8.24) e a segunda (Figura 8.25)?

8-21 Quando transmitimos áudio/vídeo armazenado na forma de um fluxo contínuo (*streaming*), qual é a diferença entre a segunda abordagem (Figura 8.24) e a terceira (Figura 8.25)?

8-22 Na quarta abordagem para transmissão em fluxo contínuo (*streaming*) de áudio/vídeo, qual é o papel do RTSP?

8-23 Qual é a principal diferença entre o áudio/vídeo ao vivo e o áudio/vídeo interativo em tempo real?

8-24 Quando transmitimos áudio/vídeo sob demanda, qual destes três tipos de comunicação multimídia ocorre: fluxo contínuo de áudio/vídeo armazenado, fluxo contínuo de áudio/vídeo ao vivo ou áudio/vídeo interativo em tempo real?

8-25 Na Figura 8.26, o servidor Web e o servidor de mídia podem ser executados em máquinas diferentes?

8-26 Em uma transmissão de áudio/vídeo interativo em tempo real, o que acontece se um pacote chegar no receptor após o instante programado para sua reprodução?

8-27 Considere que queiramos enviar uma palavra de dados de dois *bits*. As palavras de código a seguir podem ser utilizadas para corrigir um erro de um único *bit*?

00000 01011 10101 11110

8-28 Na Figura 8.33, o entrelaçamento funciona se houver mais do que um pacote perdido?

8-29 O esquema da Figura 8.34 ainda funciona se houver mais do que um pacote perdido?

8-30 Considere que elaboramos um protocolo com um tamanho de pacote tão grande que ele é capaz de transportar todos os blocos de um fluxo contínuo (*streaming*) multimídia ao vivo ou em tempo real em um único pacote. Ainda assim, precisamos de números sequência ou carimbos de tempo para os pedaços? Explique.

8-31 Explique por que o RTP não pode ser usado como um protocolo da camada de transporte sem ser executado sobre outro protocolo da camada de transporte, tal como o UDP.

8-32 Tanto o TCP como o RTP usam números de sequência. Os números de sequência nesses dois protocolos desempenham o mesmo papel? Explique.

8-33 É possível fornecer um serviço adequado para aplicações multimídia interativas em tempo real usando apenas o UDP, sem o RTP?

8-34 Se capturarmos pacotes RTP, na maioria das vezes vemos que o tamanho total do cabeçalho RTP é de 12 *bytes*. Você consegue explicar esse fato?

8-35 A codificação e a decodificação dos dados multimídia são feitas pelo RTP? Explique.

8-36 Considere que uma imagem seja enviada da origem para o destino usando 10 pacotes RTP. É possível fazer com que os cinco primeiros pacotes definam que a codificação é JPEG e que os últimos cinco pacotes definam que ela é GIF?

8-37 O UDP não cria uma conexão. Como os diferentes blocos de dados, transportados em diferentes pacotes RTP, são correlacionados?

8-38 Considere que um aplicativo utilize fluxos separados para o áudio e para o vídeo durante uma sessão RTP. Quantos SSRCs e CSRCs são usados em cada pacote RTP?

8-39 Podemos dizer que a combinação do UDP com o RTP equivale ao TCP?

8-40 Por que o RTP requer o serviço de outro protocolo, o RTCP, mas o mesmo não acontece com o TCP?

8-41 O SIP precisa usar os serviços do RTP? Explique.

8-42 Mencionamos que o SIP é um programa da camada de aplicação usado para fornecer um mecanismo de sinalização entre o usuário que faz a chamada e aquele que a recebe. Qual parte nessa comunicação é o servidor e qual é o cliente?

8-43 Neste capítulo, discutimos o uso do SIP para comunicações de áudio. Existe alguma desvantagem que impeça seu uso para comunicações de vídeo?

8-44 Considere que duas partes precisem se comunicar via telefonia IP usando os serviços do RTP. Como elas podem definir os dois números de portas efêmeras a serem utilizados pelo RTP, um para cada direção da comunicação?

8-45 Em qual situação, em uma sessão *unicast* ou em uma *multicast*, as informações de retorno sobre a sessão provenientes de um pacote RTCP podem ser tratadas mais facilmente pelo emissor?

8-46 A combinação de RTP/RTCP e SIP pode ser utilizada em um ambiente sem fio? Explique.

8-47 Pesquise e descubra se o SIP pode fornecer os seguintes serviços encontrados em aparelhos telefônicos modernos.

a. Identificador de chamadas

b. Colocar chamadas em espera

c. Chamada com múltiplos usuários

8-48 Na telefonia via Internet, explique como uma chamada proveniente de Alice pode ser direcionada para Bob quando ele estiver em seu escritório ou em casa.

8-49 Você acredita que o H.323 é, na realidade, equivalente ao SIP? Quais são as diferenças? Compare as duas soluções.

8-50 O H.323 também pode ser usado para vídeo?

8-51 Em uma agência bancária, há dois atendentes nos caixas. Um presta atendimento exclusivamente aos clientes empresariais; o outro presta atendimento a clientes comuns. Esse é um exemplo de fila de prioridade? Explique.

8-52 Em uma agência bancária, para encurtar as filas de clientes, o gerente criou três filas. Se houver apenas uma atendente nos caixas atendendo um cliente de cada fila, que tipo de esquema de filas temos aqui?

8-53 Em uma agência bancária, há apenas um atendente nos caixas, mas duas linhas de clientes: empresariais e comuns. O atendente presta atendimento aos clientes comuns somente se não houver clientes empresariais esperando. Que tipo de esquema de filas temos aqui? Explique.

8-54 Em uma agência bancária, há apenas um atendente nos caixas, mas duas filas de clientes: empresariais e comuns. O atendente presta atendimento a dois clientes da fila de empresarial para cada cliente atendido na fila comum. Que tipo de esquema de filas temos aqui?

Problemas

8-1 Dada a mensagem a seguir, determine os dados comprimidos correspondentes usando codificação RLE (*Run-Length Encoding*).

AAACCCCCCBCCCCDDDDDAAAABBB

8-2 Dada a mensagem a seguir, determine os dados comprimidos correspondentes usando a segunda versão da codificação RLE com a contagem representada por um número binário de quatro *bits*.

1000000100000100000000000000010000001

8-3 Na codificação baseada em dicionário, você consegue determinar facilmente o código se a mensagem for uma das seguintes? Considere que o alfabeto da mensagem tem apenas um caractere.

a. "A" **d.** "AAAA"

b. "AA" **e.** "AAAAA"

c. "AAA" **f.** "AAAAAA"

8-4 Usando a codificação LZW, dada a mensagem "AACCCBCCDDAB":

a. Codifique a mensagem. (Ver Figura 8.2.)

b. Determine a taxa de compressão se usarmos 8 *bits* para representar um caractere e 4 *bits* para representar um dígito (hexadecimal).

8-5 Considere o código "0026163301" na codificação LZW. Supondo que o alfabeto é composto por quatro caracteres: "A", "B", "C" e "D", decodifique a mensagem (Ver Figura 8.3.).

8-6 Dada a mensagem "**AACCCBCCDDAB**", na qual as probabilidades dos símbolos são P(A) = 0,50, P(B) = 0,25, P(C) = 0,125 e P(D) = 0,125:

a. Codifique os dados usando codificação de Huffman.

b. Determine a taxa de compressão se cada caractere original for representado por 8 *bits*.

8-7 Na codificação de Huffman, é dada a seguinte tabela de codificação.

$A \rightarrow 0 \quad B \rightarrow 10 \quad C \rightarrow 110 \quad D \rightarrow 111$

Mostre a mensagem original correspondente ao código "00110110011110111111010".

8-8 Considere a mensagem "**ACCBCAAB***", na qual as probabilidades dos símbolos são P(A) = 0,4, P(B) = 0,3, P(C) = 0,2 e P(*) = 0,1.

a. Determine os dados comprimidos usando codificação aritmética com uma precisão de 10 dígitos binários.

b. Determine a taxa de compressão se usarmos 8 *bits* para representar um caractere na mensagem.

8-9 Na codificação aritmética, considere que recebemos o código 100110011. Se o alfabeto for composto por quatro símbolos com as probabilidades P(A) = 0,4, P(B) = 0,3, P(C) = 0,2 e P(*) = 0,1, determine a mensagem original.

8-10 Na codificação preditiva, considere que amostramos o seguinte (x_n).

n	1	2	3	4	5	6	7	8	9	10	11
x_n	13	24	46	60	45	32	30	40	30	27	20

a. Mostre a mensagem codificada que será enviada se usarmos a modulação delta (DM). Considere $y_0 = 10$ e $\Delta = 8$.

b. A partir dos valores de q_n calculados, o que você pode dizer sobre o Δ dado?

8-11 Na codificação preditiva, temos o código a seguir. Mostre como podemos calcular o valor reconstruído (y_n) para cada amostra se usarmos modulação delta (DM). Considere $y_0 = 8$ e $\Delta = 6$.

n	1	2	3	4	5	6	7	8	9	10	11
C_n	1	0	0	1	0	1	1	0	0	1	1

8-12 Na codificação preditiva, considere que temos a seguinte amostra (x_n). Mostre a mensagem codificada enviada se usarmos DM adaptativa (ADM). Considere $y_0 = 10$, $\Delta_1 = 4$ e $M_1 = 1$. Assuma também que $M_n = 1,5 \times M_{n-1}$ se $q_n = q_{n-1}$ (nenhuma mudança em q_n) e $M_n = 0,5 \times M_{n-1}$ caso contrário.

n	1	2	3	4	5	6	7	8	9	10	11
x_n	13	15	15	17	20	20	18	16	16	17	18

8-13 Considere que temos o seguinte código. Mostre como podemos calcular o valor reconstruído (y_n) para cada amostra se usarmos ADM. Considere $y_0 = 20$, $\Delta_1 = 4$ e $M_1 = 1$ e também que $M_n = 1,5 \times M_{n-1}$ se $q_n = q_{n-1}$ (nenhuma mudança em q_n) e $M_n = 0,5 \times M_{n-1}$ caso contrário.

n	1	2	3	4	5	6	7	8	9	10	11
C_n	1	1	1	0	0	1	1	0	0	1	1

8-14 Na DCT unidimensional, se $N = 1$, o cálculo da transformada torna-se uma simples multiplicação. Em outras palavras, $M = T \times p$, onde T, p e M são números (valores escalares) em vez de matrizes. Qual é o valor de T nesse caso?

8-15 Na DCT, o valor de $T(m, n)$ na codificação por transformada é sempre entre –1 e 1? Explique.

8-16 Na codificação por transformada, mostre que um receptor que recebe uma matriz M é capaz de criar a matriz original p.

8-17 Calcule a matriz T para a DCT quando $N = 1$, $N = 2$, $N = 4$ e $N = 8$.

8-18 Usando uma codificação DCT unidimensional, calcule a matriz M a partir das três matrizes p seguintes (que são representadas como matrizes-linha, mas devem ser consideradas matrizes-coluna). Interprete o resultado.

$p_1 = [1\ 2\ 3\ 4\ 5\ 6\ 7\ 8] \quad p_2 = [1\ 3\ 5\ 7\ 9\ 11\ 13\ 15]$
$p_3 = [1\ 6\ 11\ 16\ 21\ 26\ 31\ 36]$

8-19 Considere que uma imagem usa uma paleta de tamanho 8 criada a partir da tabela do

JPEG (o GIF usa a mesma estratégia, mas o tamanho da paleta é 256), contendo a combinação das seguintes cores com os níveis de intensidades indicados.

Vermelho: 0 e 7 Azul: 0 e 5 Verde: 0 e 4

Determine a paleta para essa situação e responda às seguintes perguntas:

a. Quantos *bits* são enviados para cada *pixel*?

b. Quais são os *bits* enviados para os seguintes *pixels*: vermelho, azul, verde, preto, branco e magenta (combinação de vermelho e azul, sem verde)?

8-20 Considerando a primeira abordagem para transmissão em fluxo contínuo (*streaming*) de áudio/vídeo armazenado (Figura 8.24), suponha que desejamos ouvir uma música que tem 4 *megabytes* após a compressão (uma situação normal). Se nossa conexão com a Internet é feita por meio de um modem de 56 Kbps, quanto tempo precisaremos esperar até a música começar a tocar (tempo de obtenção)?

8-21 Na abordagem de FEC usando entrelaçamento, considere que cada pacote contenha 10 amostras obtidas de um trecho amostrado de música. Em vez de colocar as 10 primeiras amostras no primeiro pacote, as 10 amostras seguintes no segundo pacote, e assim por diante, o emissor coloca no primeiro pacote as amostras de numeração ímpar das primeiras 20 amostras, no segundo pacote as amostras de numeração par das primeiras 20 amostras, e assim por diante. O receptor reordena as amostras e as reproduz. Agora, considere que o terceiro pacote é perdido durante a transmissão. O que ficará faltando no lado do receptor?

8-22 Considere que desejamos enviar uma palavra de dados de dois *bits* usando FEC baseada na distância de Hamming. Mostre como a seguinte lista de palavras de dados e palavras de código pode corrigir automaticamente erros de até um *bit* durante a transmissão.

$$00 \rightarrow 00000 \quad 01 \rightarrow 01011$$
$$10 \rightarrow 10101 \quad 11 \rightarrow 11110$$

8-23 Considere que precisamos criar palavras de código que podem corrigir automaticamente erros de um *bit*. Qual deve ser o número de *bits* redundantes (r), dado o número de *bits* na palavra de dados (k)? Lembre-se de que a palavra de código deve ter $n = k + r$ *bits*, denotados $C(n, k)$. Após encontrar essa relação, determine o número de *bits* em r quando o valor de k é 1, 2, 5, 50 e 1.000.

8-24 No problema anterior, tentamos determinar o número de *bits* que precisam ser adicionados a uma palavra de dados para corrigir um erro em um único *bit*. Se precisarmos corrigir mais do que um *bit*, o número de *bits* de redundância aumenta. Qual deve ser o número de *bits* redundantes (r) para corrigir automaticamente um ou dois *bits* (não necessariamente adjacentes) em uma palavra de dados de tamanho k? Após encontrar essa relação, determine o número de *bits* em r quando o valor de k é 1, 2, 5, 50 e 1.000.

8-25 Usando as ideias dos dois últimos problemas, podemos criar uma fórmula geral para corrigir qualquer número de erros (m) em uma palavra de código de tamanho (n). Desenvolva tal fórmula. Use a combinação de n objetos tomados x a x.

8-26 Na Figura 8.34, considere que temos 100 pacotes. Criamos dois conjuntos de pacotes com resoluções alta e baixa. Cada pacote de alta resolução transporta em média 700 *bits*. Cada pacote de baixa resolução transporta em média 400 *bits*. Quantos *bits* extras estamos enviando nesse esquema por causa da FEC? Qual é a porcentagem desses *bits* extras com relação ao número total de *bits*?

8-27 Na Figura 8.31, qual é a quantidade de dados no *buffer* de reprodução em cada um dos instantes a seguir?

a. 00:00:17 **c.** 00:00:25

b. 00:00:20 **d.** 00:00:30

8-28 A Figura 8.70 mostra o tempo de geração e de chegada de dez pacotes de áudio. Responda às seguintes perguntas:

a. Se iniciarmos a reprodução do áudio no instante t_8, quais pacotes não podem ser reproduzidos?

b. Se iniciarmos a reprodução do áudio no instante t_9, quais pacotes não podem ser reproduzidos?

8-29 Dado um pacote RTP cujos oito primeiros dígitos hexadecimais são $(86032132)_{16}$, responda às seguintes perguntas:

a. Qual é a versão do protocolo RTP?

b. Foram adicionados *bytes* de enchimento (*padding*) por mecanismos de segurança?

c. Há algum cabeçalho de extensão?

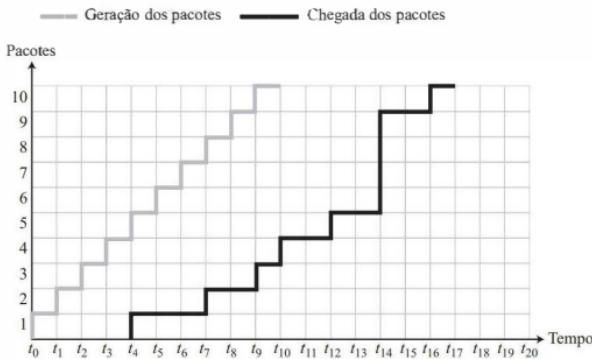



Figura 8.70 Esquema para o Problema 8-28.

d. Quantos contribuintes são definidos no pacote?

e. Qual é o tipo de carga transportada pelo pacote RTP?

f. Qual é o tamanho total do cabeçalho, em *bytes*?

8-30 Em uma comunicação multimídia em tempo real, considere que temos um emissor e dez receptores. Se o emissor estiver enviando dados multimídia a 1 Mbps, quantos pacotes RTCP podem ser enviados pelo emissor e por cada receptor em um segundo? Assuma que o sistema aloca 80% da largura de banda do RTCP para os receptores e 20% para o emissor. O tamanho médio de cada pacote RTCP é de 1.000 *bits*.

8-31 Explique por que o TCP, como um protocolo de fluxo orientado a *bytes*, não é adequado para uso em aplicações como transmissão de conteúdo multimídia ao vivo ou em tempo real.

8-32 A Figura 8.71 mostra um roteador usando filas FIFO na porta de entrada. Os instantes de chegada e os tempos necessários para processar sete pacotes são mostrados a seguir; t_i significa que o pacote chegou ou partiu i ms depois de um tempo de referência. Os valores dos tempos de processamento necessários também são mostradas em ms. Consideramos que o tempo de transmissão é desprezível.

a. Usando linhas de tempo, mostre o instante de chegada, a duração do processamento e o instante de partida de cada pacote. Mostre também o conteúdo da fila no início de cada milissegundo.

Pacotes	1	2	3	4	5	6	7
Instante de chegada	t_0	t_1	t_2	t_4	t_5	t_6	t_7
Tempo de processamento necessário	1	1	3	2	2	3	1

b. Para cada pacote, determine o tempo gasto no roteador e o atraso de partida com relação ao instante em que o pacote anterior deixou o roteador.

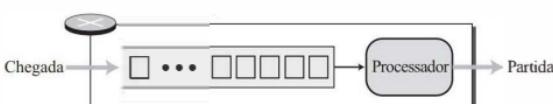



Figura 8.71 Esquema para o Problema 8-32.

c. Se todos os pacotes pertencem ao mesmo aplicativo, determinar se o roteador cria *jitter* nos pacotes.

8-33 A Figura 8.72 mostra um roteador usando filas de prioridade na porta de entrada. Os instantes de chegada e os tempos necessários para processar 10 pacotes são mostrados a seguir (o tempo de transmissão é desprezível); t_i significa que o pacote chegou i ms depois de um tempo de referência. Os valores dos tempos de processamento necessários também são mostrados em ms. Os pacotes com maior prioridade são os pacotes 1, 2, 3, 4, 7 e 9 (mostrados em cinza); os outros são pacotes com menor prioridade.

Pacotes	1	2	3	4	5	6	7	8	9	10
Instante de chegada	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
Tempo de processamento necessário	2	2	2	2	1	1	2	1	2	1

Figura 8.72 Esquema para o Problema 8-33.

a. Usando linhas de tempo, mostre o instante de chegada, a duração do processamento e o instante de partida de cada pacote. Mostre também o conteúdo da fila prioritária (F1) e da fila não prioritária (F2) a cada milissegundo.

b. Para cada pacote pertencente à classe prioritária, determine o tempo gasto no roteador e o atraso de partida com relação ao instante em que o pacote anterior deixou o roteador. Determine se o roteador cria *jitter* para essa classe.

c. Para cada pacote pertencente à classe não prioritária, determine o tempo gasto no roteador e o atraso de partida com relação ao instante em que o pacote anterior deixou o roteador. Determine se o roteador cria *jitter* para essa classe.

8-34 Para regular seu fluxo de saida, um roteador implementa um esquema de filas ponderadas com três filas na porta de saída. Os pacotes são classificados e armazenados em uma dessas filas antes de serem transmitidos. Os pesos atribuídos às filas são $w = 3$, $w = 2$ e $w = 1$ (3/6, 2/6 e 1/6). Os conteúdos de cada fila no instante t_0 são mostrados na Figura 8.73. Considere que os pacotes têm todos o mesmo tamanho e que o tempo de transmissão para cada um é 1 μs.

a. Usando uma linha de tempo, mostre o instante de partida de cada pacote.

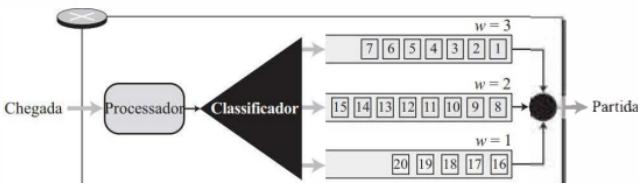



Figura 8.73 Esquema para o Problema 8-34.

b. Mostre o conteúdo das filas após 5, 10, 15 e 20 μs.

c. Determine o atraso de partida de cada pacote com relação ao pacote anterior na classe $w = 3$. O uso da fila criou *jitter* nessa classe de prioridade?

d. Determine o atraso de partida de cada pacote com relação ao pacote anterior na classe $w = 2$. O uso da fila criou *jitter* nessa classe de prioridade?

e. Determine o atraso de partida de cada pacote com relação ao pacote anterior na classe $w = 1$. O uso da fila criou *jitter* nessa classe de prioridade?

8-35 Na Figura 8.61, considere que o peso em cada classe seja 4, 2 e 1. Os pacotes na fila do alto são denotados A, na fila do meio são denotados B e na fila de baixo, C. Mostre a lista de pacotes transmitidos em cada uma das seguintes situações:

a. Cada fila tem um número grande de pacotes.

b. Os números de pacotes nas filas, de cima para baixo, são 10, 4 e 0.

c. Os números de pacotes nas filas, de cima para baixo, são 0, 5 e 10.

8-36 Em um balde furado usado para controlar o fluxo de um líquido, quantos galões de líquido sobram no balde se a taxa de saída é de 5 galões/min, há uma entrada em rajada de 100 galões/min durante 12 s e não há entrada de líquido durante 48 s?

8-37 Considere que pacotes de tamanho fixo chegam em um roteador a uma taxa de três pacotes por segundo. Mostre como o roteador pode usar o algoritmo de balde furado para enviar apenas dois pacotes por segundo. Qual é o problema com essa abordagem?

8-38 Considere que um roteador recebe pacotes cujo tamanho é de 400 *bits* a cada 100 ms, o que significa uma taxa de entrada de dados de 4 Kbps. Mostre como podemos fazer com que a taxa de dados de saída fique abaixo de 1 Kbps usando um algoritmo de balde furado.

8-39 Em um *switch* que usa o algoritmo de balde de fichas, as fichas são adicionadas ao balde a uma taxa de $r = 5$ fichas/segundo. A capacidade do balde de fichas é $c = 10$. O *switch* tem um *buffer* capaz de armazenar apenas oito pacotes (para efeitos deste exemplo). Os pacotes chegam ao *switch* a uma taxa de R pacotes/segundo. Considere que os pacotes são todos do mesmo tamanho e requerem a mesma quantidade de tempo para processamento. Se no instante zero o balde estiver vazio, mostre o conteúdo do balde e da fila em cada um dos casos a seguir e interprete o resultado.

a. $R = 5$ **b.** $R = 3$ **c.** $R = 7$

8-40 Para entender como o algoritmo de balde de fichas pode acumular crédito para um emissor que não utiliza sua taxa alocada por algum tempo, mas deseja utilizá-la mais tarde, repita o problema anterior com $r = 3$ e $c = 10$, mas considere uma taxa variável de emissão, conforme mostra a Figura 8.74. O emissor envia apenas três pacotes por segundo durante os dois primeiros segundos, não envia pacotes durante os dois segundos seguintes, e envia sete pacotes durante os próximos três segundos. O emissor tem permissão para enviar cinco pacotes por segundo, mas, como ele não usufrui desse direito totalmente durante os primeiros quatro segundos, pode enviar mais pacotes nos três segundos seguintes. Mostre o conteúdo do balde de fichas e do *buffer* em cada segundo para mostrar esse fato.

8-41 Uma interface de saída de um *switch* foi projetada usando o algoritmo de balde furado para enviar 8.000 *bytes*/s (pulso do relógio). Se os quadros a seguir forem recebidos em sequência, mostre os quadros que são enviados durante cada segundo.

- Quadros 1, 2, 3, 4: 4.000 *bytes* cada
- Quadros 5, 6, 7: 3.200 *bytes* cada

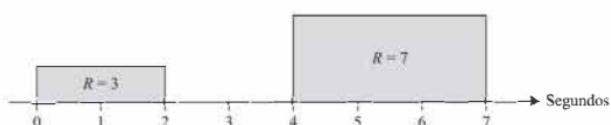



Figura 8.74 Esquema para o Problema 8-40.

- Quadros 8, 9: 400 *bytes* cada
- Quadros 10, 11, 12: 2.000 *bytes* cada

8-42 Considere que um ISP utiliza três baldes furados para regular a transmissão via Internet dos dados recebidos de três clientes. Os clientes enviam pacotes de tamanho fixo (células). Para cada cliente, o ISP envia 10 células por segundo e cria rajadas com um tamanho máximo de 20 células por segundo. Cada balde furado é implementado como uma fila FIFO (de tamanho 20) e tem um temporizador que extrai uma célula da fila e a envia a cada 1/10 de segundo. Ver Figura 8.75.

a. Mostre a taxa de transferência e o conteúdo da fila para o primeiro cliente, que envia 5 células por segundo nos primeiros 7 segundos e 15 células por segundo durante os 9 segundos seguintes.

b. Faça o mesmo para o segundo cliente, que envia 15 células por segundo nos primeiros 4 segundos e 5 células por segundo durante os 14 segundos seguintes.

c. Faça o mesmo para o terceiro cliente, que não envia células nos primeiros 2 segundos, envia 20 células nos dois segundos seguintes e então repete esse padrão quatro vezes.

8-43 Considere que o ISP do problema anterior decidiu usar baldes de fichas (capacidade $c = 20$ e taxa $r = 10$), em vez de baldes furados para permitir o acúmulo de crédito por clientes que não enviam células por algum tempo, mas precisam enviar algumas rajadas mais tarde. Cada balde de fichas é implementado por uma longa fila para cada cliente (nenhum pacote é descartado), um balde que armazena as fichas e o temporizador que regula a entrada de fichas no balde. Ver Figura 8.76.

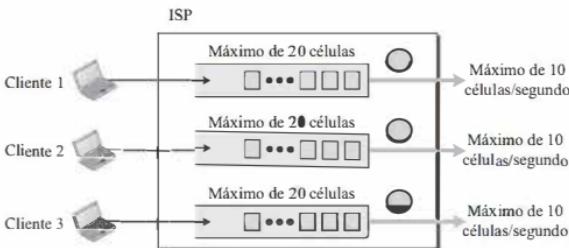



Figura 8.75 Esquema para o Problema 8-42.

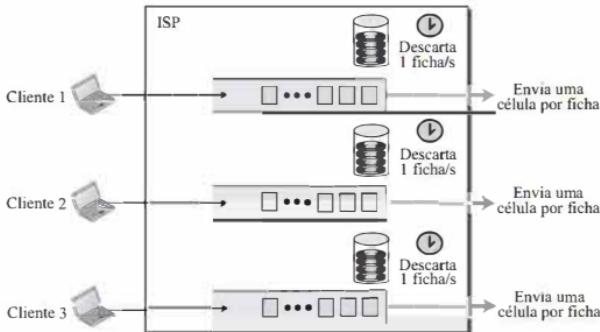



Figura 8.76 Esquema para o Problema 8-43.

a. Mostre a taxa de transferência, o conteúdo da fila e o conteúdo do balde para o primeiro cliente, que envia 5 células por segundo nos primeiros 7 segundos e 15 células por segundo durante os 9 segundos seguintes.

b. Faça o mesmo para o segundo cliente, que envia 15 células por segundo nos primeiros 4 segundos e 5 células por segundo durante os 14 segundos seguintes.

c. Faça o mesmo para o terceiro cliente, que não envia células nos primeiros 2 segundos, envia 20 células nos dois segundos seguintes e então repete esse padrão quatro vezes.

8.8 EXPERIMENTOS DE SIMULAÇÃO

8.8.1 *Applets*

Criamos alguns *applets* Java para demonstrar alguns dos principais conceitos discutidos neste capítulo. Recomendamos que o estudante ative esses *applets* no *site* www.mhhe.com/forouzan e examine cuidadosamente os protocolos em ação.

8.8.2 Experimentos de laboratório

Lab8-1 Neste experimento, precisamos examinar o conteúdo de um pacote multimídia para determinar a codificação usada pelo emissor.

Lab8-2 Neste experimento, precisamos examinar o conteúdo de um pacote multimídia para determinar qual protocolo da camada de transporte é usado na comunicação.

8.9 TAREFAS DE PROGRAMAÇÃO

Escreva o código-fonte, compile e teste os seguintes programas na linguagem de programação de sua preferência:

Prg8-1 Modifique o programa cliente-servidor UDP genérico do Capítulo 2 (em C) ou aquele do Capítulo 11 (em Java) para incluir o RTP. O programa precisa usar a biblioteca RTP (em C) ou a classe RTP (em Java) para enviar e receber imagens JPEG.

Prg8-2 Um programa para simular o balde furado (*leaky bucket*).

Prg8-3 Um programa para simular um balde de fichas (*token bucket*).

Prg8-4 Um programa para fazer a codificação e a decodificação da primeira versão do método de compressão RLE.

Prg8-5 Um programa para fazer a codificação e a decodificação da segunda versão do método de compressão RLE.

Prg8-6 Um programa que simula a Tabela 8.1 (codificação LZW).

Prg8-7 Um programa que simula a Tabela 8.2 (decodificação LZW).

Prg8-8 Um programa que simula a Tabela 8.4 (codificação aritmética).

Prg8-9 Um programa que simula a Tabela 8.5 (decodificação aritmética).

Prg8-10 Um programa que lê uma matriz bidimensional de tamanho N×N e escreve os valores ordenados em zigue-zague, conforme descrito neste capítulo.

Prg8-11 Um programa que calcula a transformada DCT de uma matriz unidimensional. Implemente a multiplicação de matrizes.

Prg8-12 Um programa que calcula a transformada DCT de uma matriz bidimensional. Implemente a multiplicação de matrizes.

CAPÍTULO

9 GERENCIAMENTO DE REDES

Embora o gerenciamento de redes seja implementado na camada de aplicação da pilha de protocolos TCP/IP, adiamos essa discussão até agora para abordá-la com mais detalhes neste capítulo. O gerenciamento de redes desempenha um papel importante na Internet à medida que ela cresce cada vez mais. A falha de um único dispositivo pode interromper a comunicação de um ponto a outro da Internet. Neste capítulo, abordamos primeiramente as áreas do gerenciamento de redes. Discutimos, em seguida, como uma dessas áreas é implementada na camada de aplicação da pilha de protocolos TCP/IP. Dividimos este capítulo em três seções:

- Na primeira seção, apresentamos o conceito de gerenciamento de redes e discutimos cinco áreas gerais do gerenciamento de redes: configuração, falhas, desempenho, segurança e contabilização. O *gerenciamento de configuração* está relacionado com o estado de cada entidade e o seu relacionamento com outras entidades. O *gerenciamento de falhas* é a área do gerenciamento de redes que lida com questões relacionadas a interrupções no sistema. O *gerenciamento de desempenho* busca monitorar e controlar a rede para garantir que ela funcione da maneira mais eficiente possível. O *gerenciamento de segurança* é responsável por controlar o acesso à rede com base em uma política predefinida. O *gerenciamento de contabilização* consiste em controlar o acesso dos usuários aos recursos de rede por meio de tarifas.
- Na segunda seção, discutimos o Protocolo Simples de Gerenciamento de Rede (SNMP – Simple Network Management Protocol) como uma estrutura para gerenciar dispositivos em uma rede usando a pilha de protocolos TCP/IP. Mostramos como um gerente, atuando como um *host*, executa um cliente SNMP, e quaisquer agentes, como roteadores ou *hosts*, executam um programa-servidor. Definimos os três componentes do protocolo de gerenciamento na Internet. Definimos também a Estrutura de Gerenciamento da Informação (SMI – Structure of Management Information), como a linguagem que especifica como tipos de dados e objetos do SNMP devem ser identificados. Em seguida, apresentamos a Base de Informações de Gerenciamento (MIB – Management Information Base), que designa os objetos a serem gerenciados no SNMP de acordo com as regras definidas na SMI.
- Na terceira seção, apresentamos uma breve discussão sobre um padrão que fornece os métodos e as regras para definir dados e objetos. Esta seção é muito breve e apenas introduz o assunto. Parte dessa discussão é usada pela SMI na segunda seção.

9.1 INTRODUÇÃO

Podemos definir o *gerenciamento de redes* como a tarefa de testar, monitorar, configurar e resolver problemas dos componentes de rede com o objetivo de atender um conjunto de requisitos definidos por uma organização. Esses requisitos incluem a operação regular e eficiente da rede, proporcionando a qualidade de serviço predefinida para os usuários. Para realizar essa tarefa, um sistema de gerenciamento de rede usa *hardware*, *software* e seres humanos.

A Organização Internacional de Padronização (ISO – International Organization for Standardization) define cinco áreas de gerenciamento de redes*: gerenciamento de configuração, de falhas, de desempenho, de segurança e de contabilização, conforme mostra a Figura 9.1.

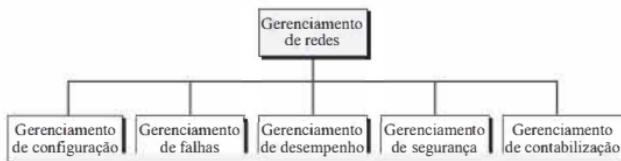



Figura 9.1 Áreas do gerenciamento de redes.

Embora algumas organizações incluam outras áreas como gerenciamento de custos, acreditamos que a taxonomia ISO é específica para o gerenciamento de redes. Por exemplo, o gerenciamento de custos é uma área de gerenciamento geral, que serve para qualquer sistema de gerenciamento e não apenas para o gerenciamento de redes.

9.1.1 Gerenciamento de configuração

Uma rede de grande porte é geralmente composta por centenas de entidades que são física ou logicamente conectadas umas às outras. Essas entidades apresentam uma configuração inicial quando a rede é inicializada, mas tal configuração pode mudar com o tempo. Os computadores pessoais podem ser substituídos por outros, um *software* de aplicação pode ser atualizado para uma versão mais recente e os usuários podem migrar de um grupo para outro. O sistema de *gerenciamento de configuração* deve saber, em qualquer momento, o estado de cada entidade e o seu relacionamento com as outras entidades. O gerenciamento de configuração pode ser dividido em dois subsistemas: *reconfiguração* e *documentação*.

Reconfiguração

A reconfiguração pode ser um evento diário em uma rede de grande porte. Existem três tipos de reconfiguração: *reconfiguração de hardware, reconfiguração de software* e *reconfiguração de conta de usuário*.

Reconfiguração de *hardware*

A reconfiguração de *hardware* envolve todas as alterações no *hardware*. Por exemplo, um computador pode precisar ser substituído. Um roteador pode precisar ser movido para outra parte da rede. Uma sub-rede pode ser adicionada ou removida da rede. Tudo isso requer o tempo e a atenção do gerenciamento de rede. Em uma rede de grande porte, deve haver pessoal especializado que seja treinado para realizar a reconfiguração de *hardware* de forma rápida e eficiente. Infelizmente, esse tipo de reconfiguração não pode ser automatizado e deve ser manuseado manualmente, caso a caso.

Reconfiguração de *software*

A reconfiguração de *software* envolve todas as alterações no *software*. Por exemplo, um novo *software* pode precisar ser instalado em servidores ou clientes. Um sistema operacional pode precisar de

* N. de T.: Essas cinco áreas também são conhecidas pela sigla FCAPS (Fault, Configuration, Accountability, Performance, and Security).

atualização. Felizmente, a reconfiguração de *software* pode ser automatizada. Por exemplo, uma atualização para um aplicativo em alguns ou todos os clientes pode ser recuperada eletronicamente, por meio de um servidor.

Reconfiguração de conta de usuário

A reconfiguração de conta de usuário não consiste simplesmente em adicionar usuários a um sistema ou excluí-los. Devemos considerar também os direitos de acesso do usuário, tanto como indivíduo quanto como membro de um grupo. Por exemplo, um usuário pode ter permissão para ler e escrever em relação a alguns arquivos, mas apenas permissão de leitura em relação a outros. A reconfiguração de conta de usuário pode ser, até certo ponto, automatizada. Por exemplo, em uma faculdade ou universidade, no início de cada trimestre ou semestre, os novos alunos são adicionados no sistema. Eles costumam ser agrupados de acordo com as disciplinas que cursam ou com as especialidades que seguem. Os membros de cada grupo têm direitos de acesso específicos; alunos de ciência da computação podem precisar acessar um servidor com funcionalidades para diferentes linguagens de computador, enquanto estudantes de engenharia podem precisar acessar servidores com *software* de Desenho Assistido por Computador (CAD – Computer Assisted Design).

Documentação

A configuração de rede original e cada alteração posterior devem ser registradas meticulosamente. Isto significa que deve haver documentações específicas para *hardware*, *software* e contas de usuário.

Documentação de *hardware*

A *documentação de hardware* normalmente envolve dois conjuntos de documentos: mapas e especificações.

Mapas Os *mapas* rastreiam cada equipamento de *hardware* e sua conexão com a rede. Pode haver um mapa geral que mostre as relações lógicas entre sub-redes. Também pode haver um segundo mapa geral que mostre a localização física de cada sub-rede. Para cada sub-rede, portanto, há um ou mais mapas que mostram todos os equipamentos. Os mapas podem usar algum tipo de padronização para serem facilmente lidos e compreendidos pelas equipes atuais e futuras.

Especificações Mapas não são suficientes por si só. Cada equipamento de *hardware* também precisa ser documentado. Deve haver um conjunto de especificações para cada equipamento de *hardware* ligado à rede. Essas especificações devem incluir informações como tipo de *hardware*, número de série, fornecedor (endereço e número de telefone), momento da compra e garantia.

Documentação de *software*

Todo *software* também deve ser documentado. A *documentação do software* inclui informações como o tipo de *software*, versão, instante de instalação e contrato de licença.

Documentação de conta de usuário

A maioria dos sistemas operacionais possui um aplicativo que permite a *documentação de conta de usuário*. O gerenciamento deve se certificar de que os arquivos com essas informações sejam atualizados e protegidos. Alguns sistemas operacionais gravam os direitos de acesso em dois documentos: um deles mostra todos os arquivos e tipos de acesso para cada usuário; o outro mostra a lista de usuários que têm acesso a um arquivo em particular.

9.1.2 Gerenciamento de falhas

Atualmente, as redes complexas são compostas por centenas e até milhares de componentes. A operação adequada da rede depende da operação adequada de cada componente individualmente e em relação uns aos outros. O *gerenciamento de falhas* é a área do gerenciamento de redes que lida com esse problema. Um sistema eficaz de gerenciamento apresenta dois subsistemas: o gerenciamento de falhas reativo e o gerenciamento de falhas proativo.

Gerenciamento de falhas reativo

Um sistema de *gerenciamento de falhas reativo* é responsável por detectar, isolar, corrigir e registrar falhas. Ele lida com soluções de curto prazo para as falhas.

Detecção de falhas

O primeiro passo tomado por um sistema de gerenciamento de falhas reativo é determinar a localização exata da falha, que é definida como uma condição anormal no sistema. Quando ocorre uma falha ocorre, o sistema deixa de funcionar corretamente ou então gera erros excessivos. Um bom exemplo de uma falha é um meio de comunicação (por exemplo, um cabo) danificado.

Isolar falhas

O passo seguinte tomado por um sistema de gerenciamento de falhas reativo é isolar a falha, que se isolada, geralmente afeta apenas alguns usuários. Após o isolamento, os usuários afetados são imediatamente notificados e informados sobre o tempo estimado de correção.

Correção de falhas

O passo seguinte é corrigir a falha. Isso pode envolver a substituição ou o reparo dos componentes defeituosos.

Registro de falhas

Após a correção da falha, ela deve ser documentada. O registro deve mostrar a localização exata da falha, a sua possível causa, a ação ou ações tomadas para corrigi-la, o custo e o tempo despendidos em cada etapa do processo. A documentação é extremamente importante por várias razões:

- O problema pode reaparecer. A documentação pode ajudar administradores e técnicos atuais ou futuros a resolver um problema semelhante.
- A frequência do mesmo tipo de falha é uma indicação de um problema maior no sistema. Se uma falha ocorre frequentemente em um componente, este deve ser substituído por um equipamento semelhante ou todo o sistema deve ser alterado para evitar o uso desse tipo de componente.
- A estatística é útil para outra área do gerenciamento de rede, o gerenciamento de desempenho.

Gerenciamento de falhas proativo

O *gerenciamento de falhas proativo* busca evitar que as falhas ocorram. Embora isso não seja sempre possível, alguns tipos de falhas podem ser previstos e evitados. Por exemplo, se um fabricante especifica um tempo de vida de um componente ou de uma parte de um componente, uma boa estratégia é substituí-lo antes que esse tempo expire. Outro exemplo é, se uma falha acontece frequentemente em um determinado ponto de uma rede, é aconselhável reconfigurar cuidadosamente a rede para evitar que a falha aconteça novamente.

9.1.3 Gerenciamento de desempenho

O *gerenciamento de desempenho,* que está intimamente relacionado ao gerenciamento de falhas, busca monitorar e controlar* a rede para garantir que ela opere da maneira mais eficiente possível. O gerenciamento de desempenho busca quantificar o desempenho usando alguma quantidade mensurável, tal como capacidade, tráfego, vazão ou tempo de resposta. Alguns protocolos, como é o caso do SNMP, que é discutido neste capítulo, podem ser usados no gerenciamento de desempenho.

Capacidade

Um fator que deve ser monitorado por um sistema de gerenciamento de desempenho é a *capacidade* da rede. Cada rede tem uma capacidade limitada e o sistema de gerenciamento de desempenho deve garantir que ela não seja usada acima dessa capacidade. Por exemplo, se uma LAN é projetada para 100 estações com uma taxa média de transferência de dados de 2 Mbps, ela não funcionará adequadamente se 200 estações forem conectadas à rede. A taxa de transferência de dados diminuirá e pode haver congestionamentos.

Tráfego

O *tráfego* pode ser medido de duas formas: interna ou externamente. O tráfego interno é medido pelo número de pacotes (ou *bytes*) que são transmitidos no interior da rede. O tráfego externo é medido pela troca de pacotes (ou *bytes*) com entidades fora da rede. Durante horários de pico, quando o sistema é muito utilizado, pode haver congestionamentos se ocorrer excesso de tráfego.

Vazão

Podemos medir a *vazão* de um dispositivo individual (um roteador, por exemplo) ou de uma parte da rede. O gerenciamento de desempenho monitora a vazão para garantir que ela não seja reduzida a níveis inaceitáveis.

Tempo de resposta

O *tempo de resposta* é normalmente medido do momento em que um usuário requisita um serviço ao momento em que o serviço é concedido. Outros fatores, como capacidade e tráfego, podem afetar o tempo de resposta. O gerenciamento de desempenho monitora o tempo de resposta médio e o tempo de resposta durante horários de pico. Qualquer elevação no tempo de resposta é um problema grave, uma vez que ele é uma indicação de que a operação da rede encontra-se acima de sua capacidade.

9.1.4 Gerenciamento de segurança

O *gerenciamento de segurança* é responsável por controlar o acesso à rede com base em uma política predefinida. No Capítulo 10, discutiremos ferramentas de segurança, como cifração e autenticação. A cifração fornece privacidade aos usuários; a autenticação obriga que os usuários se identifiquem.

---

* N. de T.: A monitoração consiste em observar e coletar informações sobre o funcionamento de um dado recurso da rede. O controle consiste em alterar o funcionamento de um elemento da rede em função dessas informações e da política de gerenciamento de redes.

9.1.5 Gerenciamento de contabilização

O *gerenciamento de contabilização* consiste no controle de acesso dos usuários aos recursos da rede por meio de tarifação. Sob a tutela do gerenciamento de contabilização, usuários individuais, departamentos, divisões ou até mesmo projetos são tarifados pelos serviços que recebem da rede. A tarifação não significa necessariamente a transferência de dinheiro; ela pode significar o débito dos departamentos ou divisões para fins orçamentários. Atualmente, as organizações utilizam um sistema de gerenciamento de contabilização pelos seguintes motivos:

- Ele impede que os usuários monopolizem os limitados recursos da rede.
- Ele impede que os usuários utilizem o sistema de forma ineficiente.
- Os gerentes de rede podem fazer planejamento de curto e longo prazo com base na demanda de uso da rede.

9.2 SNMP

Várias normas de gerenciamento de redes foram concebidas durante as últimas décadas. A mais importante é o **Protocolo Simples de Gerenciamento de Rede** (SNMP – Simple Network Management Protocol), utilizado pela Internet*. Discutimos este padrão nesta seção. O SNMP é uma estrutura para o gerenciamento de dispositivos em uma internet que use a pilha de protocolos TCP/IP**. Ele fornece um conjunto de operações fundamentais para o monitoramento e a manutenção de uma internet. O SNMP usa o conceito de gerente e agente. Isto é, um gerente, normalmente uma estação, controla e monitora um conjunto de agentes, geralmente roteadores ou servidores (Figura 9.2).

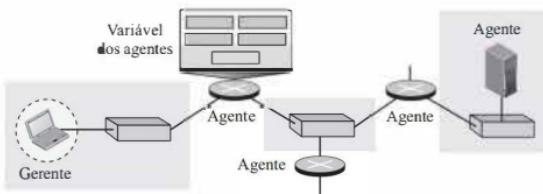



Figura 9.2 Conceito do SNMP.

O SNMP é um protocolo da camada de aplicação no qual algumas poucas estações gerentes controlam um conjunto de agentes. O protocolo foi desenvolvido na camada de aplicação para que ele possa monitorar os dispositivos produzidos por diferentes fabricantes e instalados em diferentes redes físicas. Em outras palavras, o SNMP torna as tarefas de gerenciamento independentes tanto das características físicas dos dispositivos sendo gerenciados como da tecnologia de rede subjacente. Ele pode ser usado em uma internet heterogênea composta por diferentes LANs e WANs conectadas por roteadores produzidos por diferentes fabricantes.

* N. de T.: O SNMP é geralmente usado para o gerenciamento de recursos apenas até a camada de redes. Para camadas altas, é mais comum a adoção do Gerenciamento Corporativo Baseado na Web (WBEM – Web Based Enterprise Management) ou do Gerenciamento de Serviços Web (WS-Management – Web Services Management). Mais informações sobre esses protocolos podem ser encontradas em http://dmtf.org/standards.

** N. de T.: Também é possível monitorar redes não baseadas no TCP/IP através do uso de *proxies* SNMP.

9.2.1 Gerentes e agentes

Uma estação de gerenciamento, chamado de *gerente*, é um *host* que executa o programa-cliente SNMP. Uma estação gerenciada, chamada de *agente*, é um roteador (ou uma estação) que executa o programa-servidor SNMP. O gerenciamento é obtido por meio da simples interação entre um gerente e um agente.

O agente possui acesso a informações de desempenho em uma base de dados local. O gerente tem acesso aos valores na base de dados por meio do agente. Por exemplo, um roteador pode armazenar, usando variáveis adequadas, o número de pacotes recebidos e encaminhados. O gerente pode buscar e comparar os valores dessas duas variáveis para determinar se o roteador está congestionado ou não.

O gerente também pode fazer o roteador executar determinadas ações. Por exemplo, um roteador verifica periodicamente o valor de um contador de reinício para determinar quando ele deve reiniciar a si próprio. Ele reinicia a si próprio, por exemplo, se o valor do contador for 0. O gerente pode usar esse recurso para reiniciar o agente remotamente, a qualquer momento. Ele simplesmente envia um pacote para forçar um valor 0 no contador.

Os agentes podem também contribuir com o processo de gerenciamento. O programa-servidor executando no agente pode verificar o ambiente e, se notar algo estranho, ele pode enviar uma mensagem de *alerta* (também conhecida como *trap*) para o gerente.

Em outras palavras, o gerenciamento no SNMP se baseia em três ideias básicas:

1. Um gerente inspeciona um agente por meio de uma solicitação de informações que refletem no comportamento do agente.
2. Um gerente força um agente a executar uma tarefa por meio da reinicialização de valores na base de dados daquele agente.
3. Um agente contribui com o processo de gerenciamento por meio de alertas, avisando o gerente de uma situação incomum.

9.2.2 Componentes de gerenciamento

Para realizar as tarefas de gerenciamento, o SNMP utiliza dois outros protocolos: **Estrutura de Gerenciamento da Informação** (SMI – Structure of Management Information) e **Base de Informações de Gerenciamento** (MIB – Management Information Base). Em outras palavras, o gerenciamento da Internet é efetuado por meio da cooperação entre três protocolos: SNMP, SMI e MIB, conforme mostra a Figura 9.3.

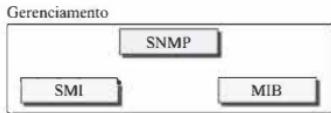



Figura 9.3 Componentes do gerenciamento de redes na Internet.

Detalharemos as interações entre esses protocolos.

Papel do SNMP

O SNMP tem alguns papéis bastante específicos no gerenciamento de redes. Ele define o formato do pacote a ser enviado de um gerente para um agente e vice-versa. Ele também interpreta o resultado

e cria estatísticas (muitas vezes com a ajuda de outro *software* de gerenciamento). Os pacotes trocados contêm os nomes dos objetos (variáveis) e seus estados (valores). O SNMP é responsável por ler e alterar esses valores.

O SNMP define o formato dos pacotes trocados entre um gerente e um agente. Ele lê e altera o estado de objetos (valores das variáveis) nos pacotes SNMP.

Papel da SMI

Para usar o SNMP, precisamos de regras para nomear os objetos. Isto é particularmente importante porque os objetos no SNMP formam uma estrutura hierárquica (um objeto pode ter um objeto-pai e alguns objetos-filhos). Parte de um nome pode ser herdada do pai. Também precisamos de regras para definir os tipos de objetos. Que tipos de objetos são manipulados pelo SNMP? O SNMP pode lidar com tipos simples ou tipos estruturados? Quantos tipos simples existem? Quais são os tamanhos desses tipos? Qual é a faixa de valores desses tipos? Além disso, como cada um desses tipos é codificado?

A SMI define as regras gerais para nomear objetos, definir os tipos de objetos (incluindo a faixa de valores e seu comprimento) e mostrar como codificar objetos e valores.

Precisamos dessas regras universais porque não sabemos a arquitetura dos computadores que enviam, recebem ou armazenam esses valores. O emissor pode ser um computador robusto no qual um número inteiro é armazenado usando 8 *bytes*; o receptor pode ser um computador de pequeno porte que armazena um número inteiro usando 4 *bytes*.

A SMI é uma especificação que define essas regras. No entanto, devemos ter em mente que a SMI apenas define as regras; ela não define quantos objetos são gerenciados em uma entidade ou qual objeto usa qual tipo. A SMI é uma coleção de regras gerais para nomear objetos e listar seus tipos. A associação de um objeto com o tipo não é feita pela SMI.

Papel da MIB

Esperamos que esteja claro que precisamos de outro componente, capaz de definir quais informações estarão disponíveis no agente. Para cada entidade a ser gerenciada, essas informações devem definir o número de objetos, nomeá-los de acordo com as regras definidas pela SMI e associar um tipo a cada objeto nomeado. Esse componente é a MIB. A MIB cria um conjunto de objetos definidos para cada entidade de maneira semelhante à feita por uma base de dados (basicamente metadados em uma base de dados, consistindo em nomes e tipos sem valores a eles atribuídos).

A MIB cria uma coleção de objetos nomeados, seus tipos e os relacionamentos entre eles em uma entidade a ser gerenciada.

Uma analogia

Antes de discutirmos cada um desses componentes em mais detalhes, faremos uma analogia. Os três componentes de gerenciamento de redes são semelhantes aos que precisamos quando escrevemos um programa em uma linguagem de computador para resolver um problema. A Figura 9.4 mostra essa analogia.

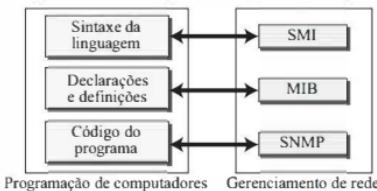



Figura 9.4 Comparando programação de computadores e gerenciamento de redes.

Sintaxe: SMI

Antes de escrevermos um programa, a sintaxe da linguagem (por exemplo, C ou Java) deve ser predefinida. A linguagem também define a estrutura das variáveis (simples, estruturada, ponteiro, e assim por diante) e como as variáveis devem ser nomeadas. Por exemplo, um nome da variável deve ter um comprimento de 1 a *n* caracteres e deve começar com uma letra seguida por caracteres alfanuméricos. A linguagem também define o tipo de dados a serem utilizados (número inteiro, número real, caracteres, etc.) Na programação, as regras são definidas pela sintaxe da linguagem. No gerenciamento de redes, as regras são definidas pela SMI.

Declaração e definição de objetos: MIB

A maioria das linguagens de computador exige que os objetos sejam declarados e seus tipos sejam definidos em cada programa específico. As declarações e definições criam objetos usando tipos predefinidos e alocam memória para eles. Por exemplo, se um programa tem duas variáveis (um inteiro chamado de *contador* e um vetor de caracteres chamado *notas*), eles devem ser declarados no início do programa:

```
int contador;
char notas [40];
```

A MIB realiza essa tarefa no gerenciamento de redes. A MIB nomeia cada objeto e define o tipo dos objetos. Como o tipo é definido pela SMI, o SNMP sabe a sua faixa de valores e o seu tamanho.

Código do programa: SNMP

Após a declaração na programação, o programa precisa apresentar as instruções que armazenam valores nas variáveis e modificam esses valores conforme for a necessidade. O SNMP realiza essa tarefa no gerenciamento de redes. O SNMP armazena, modifica e interpreta os valores dos objetos já declarados pela MIB, de acordo com as regras definidas pela SMI.

9.2.3 Uma visão geral

Antes de discutir cada componente em mais detalhes, mostraremos o papel de cada um desses componentes em um cenário simples. É uma visão geral que será desenvolvida mais adiante, no final do capítulo. Uma estação-gerente (cliente SNMP) deseja enviar uma mensagem para uma estação-agente (servidor SNMP) com o objetivo de determinar o número de datagramas de usuário UDP recebidos pelo agente. A Figura 9.5 mostra uma visão geral das etapas envolvidas.

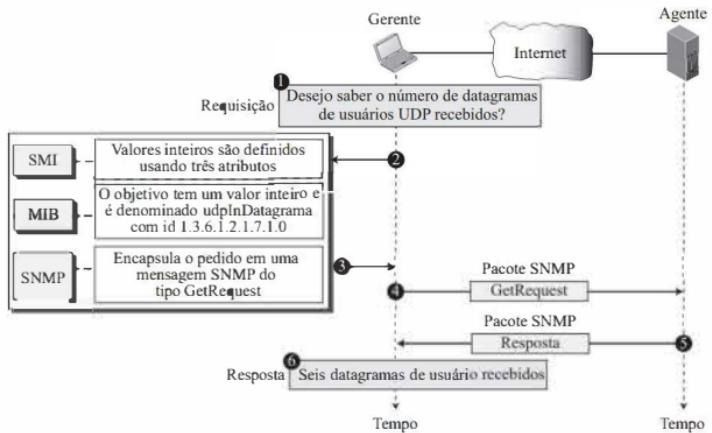



Figura 9.5 Visão geral de gerenciamento.

A MIB é responsável por encontrar o objeto que contém o número de datagramas de usuário UDP recebidos. A SMI, com a ajuda de outro protocolo a ela incorporado, é responsável por codificar o nome do objeto. O SNMP é responsável por criar uma mensagem, conhecida como "GetRequest", e por encapsular a mensagem codificada. Obviamente, as operações são mais complicadas do que nessa simples visão geral, mas precisamos primeiro de mais detalhes sobre cada protocolo.

9.2.4 SMI

A Estrutura de Gerenciamento de Informação, versão 2 (SMIv2) é um componente do gerenciamento de redes. A SMI é uma diretriz usada pelo SNMP. Ela define três atributos para tratar um objeto: nome, tipo de dados e método de codificação. Suas funções são:

- Nomear objetos.
- Definir o tipo de dado que pode ser armazenado dentro de um objeto.
- Mostrar como codificar dados para transmissão através da rede.

Nome

A SMI exige que cada objeto gerenciado (por exemplo, um roteador, uma variável em um roteador, um valor, etc.) tenha um nome único. Para nomear objetos globalmente, a SMI usa um **identificador de objeto**, que consiste em um identificador hierárquico baseado em uma estrutura em árvore (ver Figura 9.6).

A estrutura da árvore começa com uma raiz sem nome. Cada objeto pode ser definido usando uma sequência de inteiros separados por pontos. A estrutura da árvore também pode definir um objeto usando uma sequência de nomes textuais separados por pontos.

A representação inteiro-ponto é usada pelo SNMP. A notação inteiro-ponto é usada por pessoas. O exemplo a seguir mostra o mesmo objeto representado por meio de duas notações diferentes.

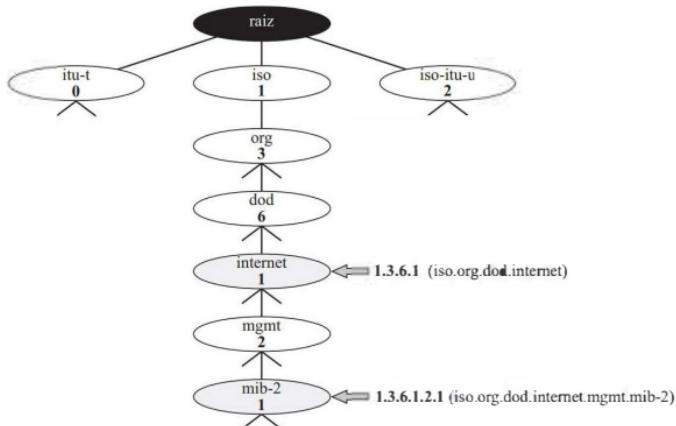



Figura 9.6 Identificador de objeto na SMI.

Os objetos que são usados pelo SNMP ficam localizados abaixo do objeto mib-2, de modo que seus identificadores sempre começam com 1.3.6.1.2.1.

Tipo

O segundo atributo de um objeto é o tipo de dados armazenados nele. Para definir o tipo de dados, a SMI usa as definições da **Notação Sintática Abstrata Um** (ASN.1 – Abstract Syntax Notation One), e acrescenta algumas novas definições. Em outras palavras, a SMI é tanto um subconjunto como um superconjunto do ASN.1. Discutiremos o ASN.1 na última seção deste capítulo.

A SMI apresenta duas grandes categorias de tipos de dados: *simples* e *estruturada*. Começamos definindo os tipos simples e então mostramos como os tipos estruturados podem ser construídos a partir dos simples.

Tipo simples

Os **tipos de dados simples** são tipos de dados atômicos. Alguns deles vêm diretamente do ASN.1, enquanto outros foram adicionados pela SMI. Os tipos mais importantes são mostrados na Tabela 9.1. Os cinco primeiros vêm do ASN.1; os sete seguintes são definidos pela SMI.

Tabela 9.1 Tipos de dados.

Tipo	Tamanho	Descrição
INTEGER	4 *bytes*	Um inteiro com valor entre -2^{31} e $2^{31}-1$
Integer32	4 *bytes*	Igual ao INTEGER

(Continua)

Tabela 9.1 Tipos de dados. (*Continuação*)

Tipo	Tamanho	Descrição
Unsigned32	4 *bytes*	Inteiro sem sinal com valor entre 0 e $2^{32}-1$
OCTET STRING	Variável	Sequência de *bytes* com até 65.535 *bytes* de comprimento
OBJECT IDENTIFIER	Variável	Um identificador de objeto
IPAddress	4 *bytes*	Um endereço IP composto por quatro inteiros
Counter32	4 *bytes*	Um contador na forma de um inteiro cujo valor pode ser incrementado de zero até 2^{32}; quando ele atinge seu valor máximo, volta a valer zero
Counter64	8 *bytes*	Um contador de 64 *bits*, valendo de zero a 2^{64}
Gauge32	4 *bytes*	Equivalente ao Counter32, mas quando ele atinge seu valor máximo, não retorna a zero; ele permanece nesse valor até ser reiniciado
TimeTicks	4 *bytes*	Um valor de contagem incrementado em intervalos 1/100 segundos
BITS	Variável	Uma sequência de *bits*
Opaque	Variável	O tipo opaco consiste em uma sequência de *bits* não interpretada

Tipo estruturado

Por meio da combinação de tipos de dados simples e estruturados, podemos criar novos tipos de dados estruturados. A SMI define dois **tipos de dados estruturados**: *sequência* e *sequência de*.

- **Sequência.** Um tipo de dados de *sequência* consiste em uma combinação de tipos de dados simples, não necessariamente do mesmo tipo. Esse tipo de dados é análogo ao conceito de *struct* (estrutura) ou *record* (registro) usado em linguagens de programação, como C.
- **Sequência de.** Um tipo de dados denominado *sequência de* é uma combinação de tipos de dados simples, todos do mesmo tipo, ou uma combinação de tipos de dados de sequência, todos do mesmo tipo. Esse tipo de dados é análogo ao conceito de um vetor usado em linguagens de programação, como C.

A Figura 9.7 mostra uma visão conceitual dos tipos de dados.

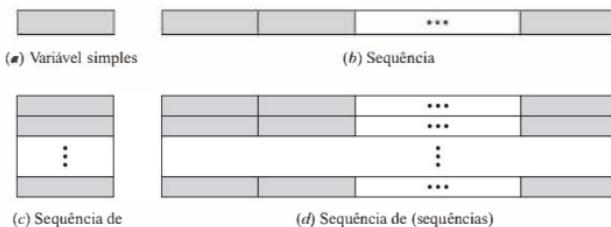



Figura 9.7 Visão conceitual dos tipos de dados.

Método de codificação

A SMI usa outro padrão, **o Regras Básicas de Codificação** (BER – Basic Encoding Rules), para codificar os dados a serem transmitidos pela rede. O BER especifica que cada dado seja codificado como uma tripla: rótulo, comprimento e valor (*tag-length-value*, ou TLV), como ilustrado na Figura 9.8.

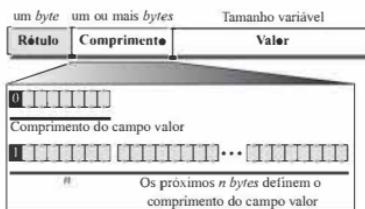



Figura 9.8 Formato da codificação.

O rótulo é um campo de 1 *byte* que define o tipo dos dados. A Tabela 9.2 mostra os tipos de dados que utilizamos neste capítulo e seus respectivos rótulos na forma de números hexadecimais. O campo de comprimento apresenta 1 ou mais *bytes*. Se ele possuir apenas um *byte*, o *bit* mais significativo deve ser 0. Os outros 7 *bits* definem o comprimento dos dados. Se ele possuir mais do que um *byte*, o *bit* mais significativo do primeiro *byte* deve ser 1. Os outros 7 *bits* do primeiro *byte* especificam o número de *bytes* necessários para definir o comprimento. O campo de valor codifica o valor dos dados de acordo com as regras definidas no BER.

Tabela 9.2 Códigos para tipos de dados.

Tipo de dado	Rótulo (Hex)	Tipo de dado	Rótulo (Hex)
INTEGER	**02**	IPAddress	**40**
OCTET STRING	**04**	Counter	**41**
OBJECT IDENTIFIER	**06**	Gauge	**42**
NULL	**05**	TimeTicks	**43**
SEQUÊNCIA, SEQUÊNCIA DE	**30**	Opaque	**44**

Exemplo 9.1

A Figura 9.9 mostra como definir um inteiro (INTEGER) de valor 14. O tamanho do campo de comprimento é dado na Tabela 9.1

0x02	0x04	0x00	0x00	0x00	0x0E
Rótulo (INTEGER)	Comprimento (4 *bytes*)	Valor (14)			

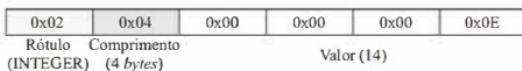



Figura 9.9 Esquema para o Exemplo 9.1: Inteiro (INTEGER) de valor 14.

Exemplo 9.2

A Figura 9.10 mostra como definir uma sequência de *bytes* (OCTET STRING) com o valor "HI".

0x04	0x02	0x48	0x49
Rótulo (OCTET STRING)	Comprimento (2 *bytes*)	Valor (H)	Valor (I)

Figura 9.10 Esquema para o Exemplo 9.2: Sequência de *bytes* (OCTET STRING) com o valor "HI".

Exemplo 9.3

A Figura 9.11 mostra como definir o identificador de objeto (OBJECT IDENTIFIER) de valor 1.3.6.1 (iso.org.dod. internet).

0x06	0x04	0x0I	0x03	0x06	0x01
Rótulo (OBJECT ID)	Comprimento (4 *bytes*)	Valor (1)	Valor (3)	Valor (6)	Valor (1)
		← 1.3.6.1 (iso.org.dod.internet) →			

Figura 9.11 Esquema para o Exemplo 9.3: Identificador de objeto (OBJECT IDENTIFIER) 1.3.6.1.

Exemplo 9.4

A Figura 9.12 mostra como definir um endereço IP (IPAddress) de valor 131.21.14.8.

0x40	0x04	0x83	0x15	0x0E	0x08
Rótulo (IPAddress)	Comprimento (4 *bytes*)	Valor (131)	Valor (21)	Valor (14)	Valor (8)
		← 131.21.14.8 →			

Figura 9.12 Esquema para o Exemplo 9.4: Endereço IP (IPAddress) 131.21.14.8.

9.2.5 MIB

A Base de Informações de Gerenciamento versão 2 (MIB2) é o segundo componente utilizado no gerenciamento de redes. Cada agente tem sua própria MIB2, que é uma coleção de todos os objetos que o gerente pode gerenciar (ver Figura 9.13).

Os objetos na mib-2 são categorizados em vários grupos: sistema, interface, tradução de endereços, ip, icmp, tcp, udp, egp, transmissão e snmp (note que o grupo 9 é obsoleto). Esses grupos estão abaixo do objeto mib-2 na árvore de identificadores de objeto. Cada grupo define variáveis e/ou tabelas.

Uma breve descrição de alguns dos objetos é fornecida a seguir:

- **sys** Objeto (*system*, ou *sistema*) que define informações gerais sobre o nó (sistema), como nome, localização e tempo de vida.

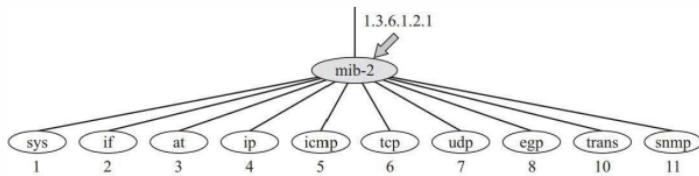



Figura 9.13 Alguns grupos do mib-2.

- **if** Objeto (*interface*) que define informações sobre todas as interfaces do nó, incluindo o número de interface, endereço físico e endereço IP.
- **at** Objeto (*address translation*, ou *tradução de endereços*) que define informações sobre a tabela ARP.
- **ip** Objeto que define informações relacionadas ao IP, como a tabela de roteamento e o endereço IP.
- **icmp** Objeto que define informações relacionadas ao ICMP, como o número de pacotes enviados e recebidos e o número total de erros criados.
- **tcp** Objeto que define informações gerais relacionadas ao TCP, como a tabela de conexões, o valor do tempo-limite, os números de portas e o número de pacotes enviados e recebidos.
- **udp** Objeto que define informações gerais relacionadas ao UDP, como os números de portas e o número de pacotes enviados e recebidos.
- **egp** Objetos relacionados à operação do EGP.
- **trans** Objetos relacionados ao método específico de transmissão (reservado para uso futuro).
- **snmp** Objeto que define informações gerais relacionadas ao próprio SNMP.

Acessando variáveis MIB

Para mostrar como acessar diferentes variáveis, usamos o grupo udp como um exemplo. Existem quatro variáveis simples no grupo udp e uma sequência de (tabela de) registros. A Figura 9.14 mostra as variáveis e a tabela. Mostraremos como acessar cada entidade.

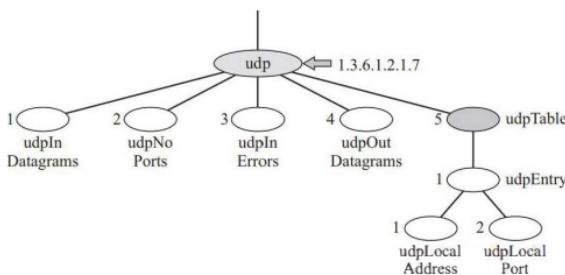



Figura 9.14 Grupo udp.

Variáveis simples

Para acessar qualquer uma das variáveis simples, usamos o identificador do grupo (1.3.6.1.2.1.7), seguido do identificador da variável. A seguir, mostraremos como acessar cada variável: número de datagramas de usuário recebidos (udpInDatagrams), número de datagramas de usuário que não foram enviados para uma porta válida (udpNoPorts), número de datagramas de usuário recebidos com erro (udpInErrors) e número de datagramas de usuário enviados (udpOutDatagrams).

udpInDatagrams	→	**1.3.6.1.2.1.7.1**
udpNoPorts	→	**1.3.6.1.2.1.7.2**
udpInErrors	→	**1.3.6.1.2.1.7.3**
udpOutDatagrams	→	**1.3.6.1.2.1.7.4**

No entanto, esses identificadores de objeto definem a variável, não a instância (conteúdo). Para mostrar a instância-exemplo ou o conteúdo de cada variável, devemos acrescentar um sufixo de instância. O sufixo de instância para uma variável simples é simplesmente um zero. Em outras palavras, para mostrar uma instância das variáveis anteriores, usamos o seguinte:

udpInDatagrams.0	→	**1.3.6.1.2.1.7.1.0**
udpNoPorts.0	→	**1.3.6.1.2.1.7.2.0**
udpInErrors.0	→	**1.3.6.1.2.1.7.3.0**
udpOutDatagrams.0	→	**1.3.6.1.2.1.7.4.0**

Tabelas

Para identificar uma tabela, primeiro usamos o identificador da tabela. O grupo udp tem apenas uma tabela (com identificador valendo 5), conforme ilustra a Figura 9.15. Então, para acessar a tabela, usamos o seguinte:

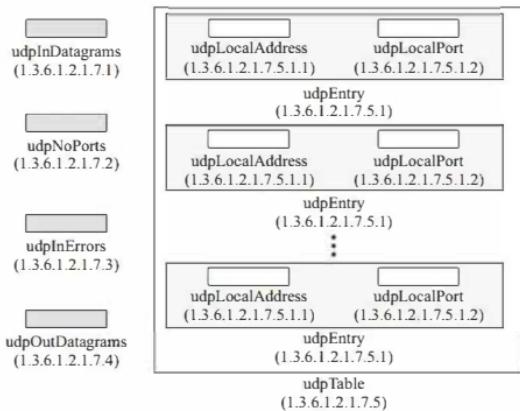



Figura 9.15 Variáveis e tabelas do udp.

No entanto, a tabela não está no nível das folhas na estrutura da árvore. Não podemos acessar a tabela; definimos a (sequência de) entrada(s) na tabela (com o identificador valendo 1), como segue:

udpEntry → **1.3.6.1.2.1.7.5.1**

Essa entrada também não é uma folha e não podemos acessá-la. Precisamos definir cada entidade (campo) na entrada da tabela.

udpLocalAddress → **1.3.6.1.2.1.7.5.1.1**
udpLocalPort → **1.3.6.1.2.1.7.5.1.2**

Essas duas variáveis estão no nível das folhas da árvore. Embora possamos acessar suas instâncias, precisamos definir *qual* é a instância. A qualquer momento, a tabela pode ter vários valores para cada par de endereço/local da porta. Para acessar uma instância específica (linha) da tabela, podemos adicionar o índice para os identificadores anteriores. Na MIB, os índices dos vetores não são números inteiros (ao contrário da maioria das linguagens de programação). Os índices são baseados no valor de um ou mais campos nas entradas. No nosso exemplo, o udpTable é indexado com base no endereço do local e no número de porta local. Por exemplo, a Figura 9.16 mostra uma tabela com quatro linhas e valores para cada campo. O índice de cada linha é uma combinação de dois valores. Para acessar a instância do endereço local da primeira linha, usamos o identificador concatenado ao índice da instância:

udpLocalAddress.181.23.45.14.23 → **1.3.6.1.2.7.5.1.1.181.23.45.14.23**

Figura 9.16 Índices para a udpTable.

9.2.6 SNMP

O SNMP usa tanto a SMI como a MIB no gerenciamento de redes Internet. Ele é um programa da camada de aplicação que permite que:

- Um gerente obtenha o valor de um objeto definido em um agente.
- Um gerente armazene um valor em um objeto definido em um agente.
- Um agente envie uma mensagem de alerta sobre uma situação anormal para o gerente.

PDUs

O SNMPv3 define oito tipos de Unidades de Dados de Protocolo (PDUs – Protocol Data Units): *GetRequest, GetNext-Request, GetBulkRequest, SetRequest, Response, Trap, InformRequest,* e *Report* (ver Figura 9.17).

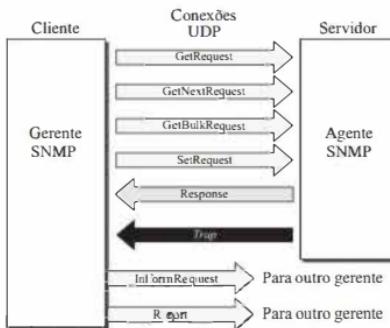



Figura 9.17 PDUs do SNMP.

GetRequest

A PDU GetRequest ("Requisição: Obter") é enviada pelo gerente (cliente) para o agente (servidor) para obter o valor de uma variável ou de um conjunto de variáveis.

GetNextRequest

A PDU GetNextRequest ("Requisição: Obter Próximo") é enviada pelo gerente para o agente para obter o valor de uma variável. O valor obtido é o do objeto de acordo com o identificador de objeto (ObjectId) definido na PDU. Essa PDU é usada principalmente para obter os valores das entradas em uma tabela. Se o gerente não sabe os índices das entradas, ele não é capaz de recuperar os valores. Entretanto, ele pode usar a GetNextRequest contendo o ObjectId da tabela. Como a primeira entrada tem o ObjectId imediatamente após o ObjectId referente à tabela, o valor da primeira entrada é retornado. O gerente pode utilizar esse ObjectId para obter o valor da próxima entrada, e assim por diante.

GetBulkRequest

A PDU GetBulkRequest ("Requisição: Obter em Massa") é enviada pelo gerente para o agente para obter uma quantidade grande de dados. Ela pode ser usada em vez de múltiplas PDUs *GetRequest* e GetNextRequest.

SetRequest

A PDU SetRequest ("Requisição: Definir") é enviada pelo gerente para o agente para definir (armazenar) um valor em uma variável.

Response

A PDU Response ("Resposta") é enviada por um agente para um gerente em resposta a um GetRequest ou GetNextRequest. Ela contém os valores das variáveis solicitadas pelo gerente.

Trap

Uma PDU de Trap ("Alerta"), também chamado de Alerta SNMPv2 para distingui-lo do Alerta SNMPv1, é enviada pelo agente para o gerente para relatar um evento. Por exemplo, se o agente for reiniciado, ele informa o gerente e informa o instante de reinicialização.

InformRequest

A PDU de InformRequest ("Requisição: Informar") é enviada por um gerente para outro gerente remoto para obter o valor de algumas variáveis dos agentes sob o controle do gerente remoto. O gerente remoto responde com uma PDU do tipo Response.

Report

A PDU de Report ("Relatório") foi projetada para relatar alguns tipos de erros entre os gerentes. Ela ainda não está em uso.

Formato

O formato das oito PDUs do SNMP é mostrado na Figura 9.18. A PDU GetBulkRequest difere das outras em duas áreas, conforme mostra a figura.

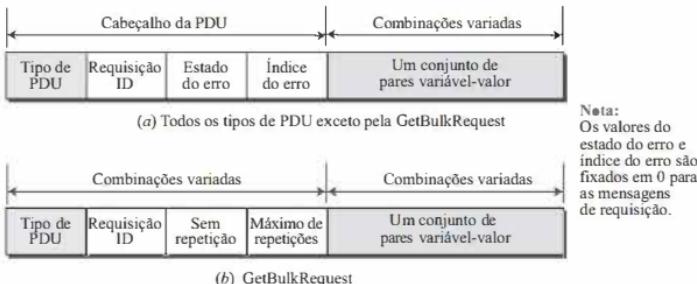



Figura 9.18 Formato da PDU SNMP.

Os campos são listados a seguir:

- **Tipo de PDU.** Esse campo define o tipo de PDU (ver Tabela 9.3).
- **ID da requisição.** É um número de sequência usado pelo gerente em uma PDU de requisição e repetido pelo agente em uma resposta. Ele é usado para identificar a resposta correspondente a um pedido.
- **Estado do erro.** É um número inteiro usado apenas em PDUs de resposta para mostrar os tipos de erros relatados pelo agente. Seu valor é 0 em PDUs de requisição. A Tabela 9.4 lista os tipos de erros que podem ocorrer.

Tabela 9.3 Tipos de PDU.

Tipo	Rótulo (Hex)	Tipo	Rótulo (Hex)
GetRequest	A0	GetBulkRequest	A5
GetNextRequest	A1	InformRequest	A6
Response	A2	Trap (SNMPv2)	A7
SetRequest	A3	Report	A8

Tabela 9.4 Tipos de erros.

Estado	Nome	Significado
0	noError	Nenhum erro
1	tooBig	Resposta é grande demais para caber em uma só mensagem
2	noSuchName	Variável não existe
3	badValue	O valor a ser armazenado é inválido
4	readOnly	O valor não pode ser modificado
5	genErr	Outros erros

- **Sem repetição.** Campo usado apenas em PDU do tipo GetBulkRequest. Ele define o número de objetos não repetidos (objetos regulares) no início da lista de pares variável-valor.
- **Índice do erro.** É um deslocamento que informa o gerente sobre qual variável causou o erro.
- **Máximo de repetições.** Campo também usado apenas em PDU GetBulkRequest. Ele define o número máximo de iterações na tabela para ler todos os objetos repetidos.
- **Lista de pares variável-valor.** Conjunto de variáveis com os valores correspondentes que o gerente deseja obter ou definir. Os valores são vazios em PDUs de requisição.

Mensagens

O SNMP não envia apenas PDUs, mas insere cada PDU em uma mensagem. Uma mensagem é composta por um cabeçalho seguido pela PDU correspondente, conforme mostra a Figura 9.19. O formato do cabeçalho da mensagem, que depende da versão e da configuração de segurança, não é mostrado na figura. Deixamos os detalhes para algum texto específico sobre o assunto.

Exemplo 9.5

Neste exemplo, uma estação gerente (cliente SNMP) usa uma mensagem com uma PDU do tipo GetRequest para obter o número de datagramas UDP que um roteador recebeu (Figura 9.20).

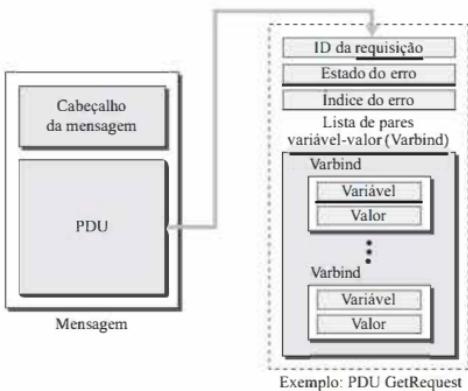



Figura 9.19 Mensagem SNMP

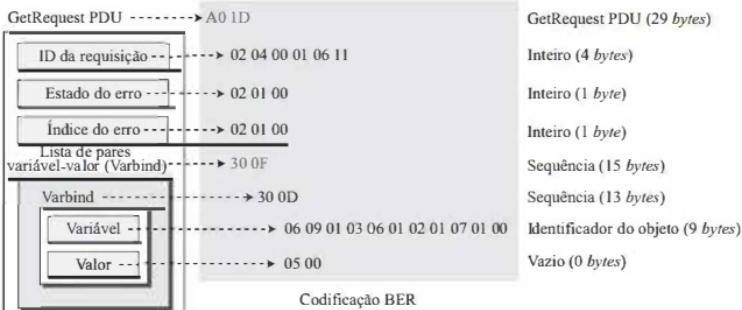



Figura 9.20 Esquema para o Exemplo 9.5.

Há apenas uma sequência de pares variável-valor, conhecidos como Varbind. A variável MIB correspondente, relacionada a essa informação, é a udpInDatagrams com o identificador do objeto 1.3.6.1.2.1.7.1.0. O gerente deseja obter um dado (não armazenar um dado), então o campo de valor define uma entidade do tipo NULL (vazio). Os *bytes* a serem enviados são mostrados na representação hexadecimal.

A lista de Varbinds tem apenas uma Varbind. A variável é do tipo 06 e seu comprimento é 09. O valor é do tipo 05 e tem comprimento 00. A Varbind completa consiste em uma sequência de comprimento 0E (13). A lista de Varbinds é também uma sequência de comprimento 0F (15). O comprimento da PDU de GetRequest é 1D (29).

Perceba que identamos os *bytes* de modo a mostrar a inclusão de tipos de dados simples dentro de uma sequência ou a inclusão de sequências e tipos de dados simples dentro de sequências maiores. Note também que a PDU em si é como uma sequência, mas seu rótulo é A0 em hexadecimal.

A Figura 9.21 mostra a mensagem de fato enviada. Consideramos que o cabeçalho da mensagem é composto por 10 *bytes*. O cabeçalho da mensagem de fato pode ser diferente. Mostramos a mensagem usando linhas de 4 *bytes*. Os *bytes* que são mostrados usando traços são aqueles relativos ao cabeçalho da mensagem.

30	29	--	--
--	--	--	--
--	--	--	--
A0	1D	02	04
00	01	06	11
02	01	00	02
01	00	30	0F
30	0D	06	09
01	03	06	01
02	01	07	01
00	05	00	

Mensagem

Nota:
Os valores dos *bytes* estão em notação hexadecimal

Figura 9.21 Mensagem de fato enviada no Exemplo 9.5.

Portas UDP

O SNMP utiliza os serviços do UDP em duas portas bem conhecidas, 161 e 162. A porta bem conhecida 161 é usada pelo servidor (agente), e a porta bem conhecida 162 é usada pelo cliente (gerente).

O agente (servidor) faz uma abertura passiva na porta 161. Em seguida, ele espera por uma conexão de um gerente (cliente), que faz uma abertura ativa usando uma porta efêmera. As mensagens de requisição são enviadas pelo cliente para o servidor usando a porta efêmera como a porta de origem e a porta bem conhecida 161 como a porta de destino. As mensagens de resposta são enviadas do servidor para o cliente usando a porta bem conhecida 161 como a porta de origem e a porta efêmera como a porta de destino.

O gerente (cliente) faz uma abertura passiva na porta 162. Em seguida, ele espera por uma conexão de um agente (servidor). Sempre que tiver uma mensagem de alerta (*trap*) para enviar, um agente (servidor) faz uma abertura ativa, usando uma porta efêmera. Essa conexão é em uma direção apenas, indo do servidor para o cliente (ver Figura 9.22).

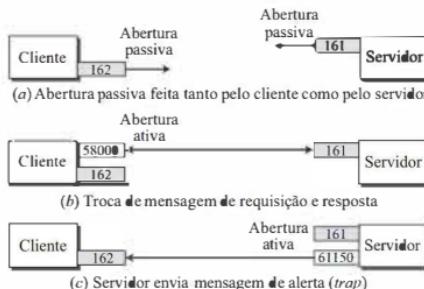



Figura 9.22 Números de porta usados pelo SNMP.

O mecanismo cliente-servidor no SNMP é diferente daquele usado em outros protocolos. Aqui, tanto o cliente como o servidor usam portas bem conhecidas. Além disso, tanto o cliente como o servidor são executados indefinidamente. A razão é que as mensagens de requisição são iniciadas por um gerente (cliente), mas as mensagens de alerta (*trap*) são iniciadas por um agente (servidor).

Segurança

O SNMPv3 acrescentou dois novos recursos à sua versão anterior: segurança e administração remota. O SNMPv3 permite que um gerente escolha um ou mais níveis de segurança quando acessa um agente. Diferentes aspectos de segurança podem ser configurados pelo gerente para permitir a autenticação, confidencialidade e integridade das mensagens.

O SNMPv3 também permite a configuração remota de aspectos de segurança sem exigir que o administrador esteja presencialmente no local onde o dispositivo está localizado.

9.3 ASN.1

Na comunicação de dados, quando enviamos um fluxo contínuo de *bits* para um destino, precisamos, de alguma forma, definir o formato dos dados. Se enviarmos um *nome* e um *número* em uma única mensagem, precisamos avisar o destino que, por exemplo, os primeiros 12 *bits* referem-se ao *nome* e os próximos 8 *bits* referem-se ao *número*. Teremos uma maior dificuldade quando enviarmos um tipo de dados complexo, tal como um vetor ou um registro. Por exemplo, no caso de um vetor, se enviarmos 2.000 *bits* em uma mensagem, precisamos informar ao receptor que os dados referem-se a um vetor de 200 números, cada um tendo 10 *bits*, ou que ele é um vetor de 10 números, cada um tendo 200 *bits*.

Uma solução é separar a definição dos tipos de dados da sequência de *bits* transmitida pela rede. Isto é feito por meio de uma linguagem abstrata que usa alguns símbolos, palavras-chave e tipos de dados atômicos, a qual nos permite criar novos tipos de dados a partir dos tipos simples. Essa linguagem é denominada Notação Sintática Abstrata Um (ASN.1 – Abstract Syntax Notation One). É importante notar que o ASN.1 é uma linguagem muito complexa utilizada em diferentes áreas da ciência da computação. Entretanto, nesta seção, apresentamos apenas os elementos da linguagem que são necessários para o protocolo SNMP.

9.3.1 Conceitos básicos da linguagem

Antes de mostrar como podemos definir objetos e valores associados, discutiremos a linguagem em si. Ela usa alguns símbolos e algumas palavras-chave, além de definir alguns tipos de dados primitivos. Como dissemos anteriormente, a SMI usa um subconjunto dessas entidades em sua própria linguagem.

Símbolos

A linguagem utiliza um conjunto de símbolos, mostrados na Tabela 9.5. Alguns desses símbolos são caracteres simples, mas alguns são pares de caracteres.

Tabela 9.5 Símbolos usados no ASN.1.

Símbolo	Significado	Símbolo	Significado
::=	Definido como ou atribuição	..	Faixa de valores

(Continua)

<table>
<tr><th colspan="4">Tabela 9.5 Símbolos usados no ASN.1. (*Continuação*)</th></tr>
<tr><th>Símbolo</th><th>Significado</th><th>Símbolo</th><th>Significado</th></tr>
<tr><td>|</td><td>Ou, alternativa, ou opção</td><td>{ }</td><td>Início e fim de uma lista</td></tr>
<tr><td>–</td><td>Sinal Negativo</td><td>[]</td><td>Início e fim de rótulo</td></tr>
<tr><td>--</td><td>O que vem a seguir é um comentário</td><td>()</td><td>Início e fim de um subtipo</td></tr>
</table>

Palavras-chave

A linguagem tem um conjunto limitado de palavras-chave que podem ser usadas. Elas podem ser utilizadas na linguagem apenas com o propósito para o qual foram definidas. Todas as palavras devem estar em letras maiúsculas (Tabela 9.6).

Tabela 9.6 Palavras-chave no ASN.1.

Palavra-chave	Descrição
BEGIN	Início de um módulo
CHOICE	Lista de alternativas
DEFINITIONS	Definição de um tipo de dados ou de um objeto
END	Fim de um módulo
EXPORTS	Tipo de dados que pode ser exportado para outros módulos
IDENTIFIER	Uma sequência de números não negativos que identifica um objeto
IMPORTS	Tipo de dados definido em um módulo externo e importado
INTEGER	Qualquer inteiro positivo, negativo ou zero
NULL	Um valor vazio
OBJECT	Usado com IDENTIFIER para definir univocamente um objeto
OCTET	Dados binários de oito *bits*
OF	Usado com SEQUENCE ou SET
SEQUENCE	Uma lista ordenada
SEQUENCE OF	Um vetor ordenado de dados do mesmo tipo
SET	Uma lista não ordenada
SETOF	Um vetor de listas não ordenadas
STRING	Uma sequência de dados

9.3.2 Tipos de dados

Agora que discutimos os símbolos e palavras-chave usados na linguagem, é hora de definir seus tipos de dados. A ideia é semelhante ao que encontramos em linguagens de programação como C, C++ ou Java. No ASN. l, temos vários tipos de dados simples, como inteiro, número em ponto flutuante, booleano, caractere e assim por diante. Podemos combinar esses tipos de dados para criar um novo tipo de dados simples (com um nome diferente) ou para definir alguns tipos de dados estruturados, como um vetor ou uma estrutura. Primeiramente, definiremos os tipos de dados simples no ASN.1; em seguida, mostraremos como criar um novo tipo de dados a partir dos tipos de dados simples.

Tipos de dados simples

O ASN.1 define um conjunto de tipos de dados simples (atômicos). Cada tipo de dados recebe um rótulo universal e apresenta um conjunto de valores, conforme mostra a Tabela 9.7. É a mesma ideia usada em uma linguagem de programação quando temos alguns tipos de dados básicos com intervalos de valores predefinidos. Por exemplo, na linguagem C, temos o tipo de dados *int*, que pode assumir uma gama de valores. Perceba que o rótulo na tabela corresponde, na realidade, aos cinco *bits* mais à direita do rótulo que definimos na Tabela 9.2.

Tabela 9.7 Alguns tipos simples-padrão do ASN.1.

Rótulo	Tipo	Conjunto de valores
Universal 1	BOOLEAN	Booleano: VERDADEIRO ou FALSO
Universal 2	INTEGER	Inteiros (positivos, 0, ou negativos)
Universal 3	BIT STRING	Uma sequência de dígitos binários (*bits*) ou uma sequência vazia
Universal 4	OCTET STRING	Uma sequência de octetos (*bytes*) ou uma sequência vazia
Universal 5	NULL	Valor único: *null* (vazio)
Universal 6	OBJECT IDENTIFIER	Um conjunto de valores que define um objeto
Universal 7	ObjectDescriptor	Texto legível para humanos descrevendo um objeto
Universal 8	EXTERNAL	Um tipo que não se encontra no padrão ASN.1
Universal 9	REAL	Números reais em notação científica
Universal 10	ENUMERATED	Uma lista de inteiros
Universal 16	SEQUENCE, SEQUENCE OF	Lista ordenada de tipos
Universal 17	SET, SET OF	Lista não ordenada de tipos
Universal 18	NumericString	Dígitos de 0-9 e espaço
Universal 19	PrintableString	Caracteres que podem ser impressos
Universal 26	VisibleString	Sequência de caracteres ISO646
Universal 27	GeneralString	Sequência genérica de caracteres
Universal 30	CHARACTER STRING	Conjunto de caracteres

Novos tipos de dados

O ASN.1 usa a sintaxe **Forma de Backus-Naur** (BNF – Backus-Naur Form) para definir um novo tipo de dados a partir de um tipo de dados padrão ou a partir de um tipo de dados previamente definido, conforme mostrado a seguir:

<NovoTipo> ::= <Tipo>

onde o *NovoTipo* deve começar com uma letra maiúscula.

Exemplo 9.6

A seguir, mostraremos um exemplo de alguns novos tipos usando tipos-padrão do ASN.1 fornecidos na Tabela 9.7.

```
Casado ::= BOOLEAN
EstadoCivil ::= ENUMERATED {solteiro, casado, viúvo, divorciado}
DiaDaSemana ::= ENUMERATED {dom, seg, ter, qua, qui, sex, sab}
Idade ::= INTEGER
```

Novos subtipos

O ASN.1 sempre permite a criação de um subtipo cuja faixa de valores seja um subintervalo de um tipo-padrão ou de um tipo de dados previamente definido.

Exemplo 9.7

O seguinte exemplo mostra como podemos criar três novos subtipos. A faixa de valores do primeiro é um subconjunto do tipo INTEGER; do segundo é um subconjunto do tipo REAL; e a do terceiro é um subconjunto de DiaDaSemana, definidos no Exemplo 9.6. Observe que usamos o símbolo (..) para definir um intervalo e o símbolo (|) para indicar alternativas.

```
NumeroDeEstudantes ::= INTEGER (15..40)     -- Um inteiro no intervalo de 15 a 40
Nota ::= REAL (1.0..5.0)                    -- Um número real no intervalo 1.0 a 5.0
FimDeSemana ::= DiaDaSemana (dom | sab)     -- Um dia que pode ser dom ou sat
```

Variáveis simples

Em uma linguagem de programação, podemos criar uma variável de um tipo particular e atribuir (armazenar) um valor a ela. No ASN.1, o termo *Valor Nome* é usado em vez da variável, mas aqui usamos o termo *variável*, que é mais familiar aos programadores. Podemos criar uma variável de um tipo particular e atribuir um valor pertencente ao intervalo definido para esse tipo. A sintaxe usada é a mostrada a seguir:

<variável> <Tipo> ::= <valor>

O nome da variável deve começar com uma letra minúscula para diferenciá-la do tipo.

Exemplo 9.8

A seguir, mostramos alguns exemplos de como definir algumas variáveis e atribuir o valor apropriado a partir da faixa de valores permitida para esses tipos. Observe que a primeira e a terceira variáveis são do tipo-padrão, a segunda é do tipo definido no Exemplo 9.6 e a última é um subtipo definido no Exemplo 9.7

```
numeroDeComputadores INTEGER ::= 2
casado Casado ::= FALSE
sualdade INTEGER ::= 35
tamanhoDaClasse NumeroDeEstudantes ::= 22
```

Tipo estruturado

O ASN.1 usa a palavra-chave SEQUENCE para definir um tipo de dados estruturado semelhante ao *struct* das linguagens C e C++. O tipo SEQUENCE é uma lista ordenada de tipos de variáveis. A seguir, mostramos um novo tipo ContaDeEstudante que consiste em uma sequência de três variáveis: nomeDeUsuario, senha e numeroDeConta.

```
ContaDeEstudante ::= SEQUENCE
{
        nomeDeUsuario VisibleString,
        senha VisibleString,
        numeroDeConta INTEGER
}
```

Variáveis estruturadas

Depois de definir o novo tipo, podemos criar uma variável a partir dele e atribuir valores a variáveis conforme mostrado a seguir:

```
joanaNewton ContaDeEstudante
{
        nomeDeUsuario "JoanaN",
        senha "120007",
        numeroDeConta 25579
}
```

A Figura 9.23 mostra o registro criado a partir da definição do tipo e atribuições de valores.

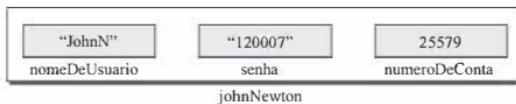



Figura 9.23 Registro representando a definição do tipo e a declaração de variável.

Usamos a palavra-chave SEQUENCE OF para definir um novo tipo semelhante a um vetor em C ou C++, que consiste em um tipo composto no qual todos os componentes são idênticos. Por exemplo, podemos definir uma tabela de roteamento em um roteador como uma SEQUENCE OF Linhas, onde cada Linha em si é uma sequência composta por diversas variáveis.

9.3.3 Codificação

Depois que os dados são definidos e os valores são associados com variáveis, o ASN.1 usa uma regra de codificação para codificar a mensagem a ser enviada. Já discutimos as Regras Básicas de Codificação (BER – Basic Encoding Rules) na seção anterior.

9.4 MATERIAL DO FINAL DO CAPÍTULO

9.4.1 Leitura adicional

Para obter mais detalhes sobre os assuntos discutidos neste capítulo, recomendamos os seguintes livros. Os itens entre colchetes referem-se à lista de referências no fim do texto.

Livros

Diversos livros fornecem uma cobertura bastante completa do SNMP: [Com 06], [Ste 94], [Tan 03] e [MS 01].

RFCs

Diversas RFCs apresentam diferentes atualizações no SNMP, incluindo RFC 3410, RFC 3412, RFC 3415 e RFC 3418. Mais informações sobre MIB podem ser encontradas em RFC 2578, RFC 2579 e RFC 2580.

9.4.2 Termos-chave

- Alerta (*trap*)
- Base de Informações de Gerenciamento (MIB – Management Information Base)
- Estrutura de Gerenciamento da Informação (SMI – Structure of Management Information)
- Forma de Backus-Naur (BNF – Backus-Naur Form)
- identificador de objeto
- Notação Sintática Abstrata Um (ASN.1 – Abstract Syntax Notation One)
- Protocolo Simples de Gerenciamento de Rede (SNMP – Simple Network Management Protocol)
- Regras Básicas de Codificação (BER – Basic Encoding Rules)
- tipo de dados estruturado
- tipo de dados simples

9.4.3 Resumo

As cinco áreas cobertas pelo gerenciamento de redes são o gerenciamento de configuração, de falhas, de desempenho, de contabilização e de segurança. O gerenciamento de configuração se preocupa com as modificações físicas ou lógicas das entidades da rede. O gerenciamento de falhas se preocupa com o funcionamento adequado de cada componente da rede. O gerenciamento de desempenho se preocupa com o monitoramento e controle da rede para garantir que ela opere da maneira mais eficiente possível. O gerenciamento de segurança se preocupa com o controle de acesso à rede. O gerenciamento de contabilização se preocupa com controlar o acesso dos usuários a recursos de rede por meio de tarifação.

O Protocolo Simples de Gerenciamento de Rede (SNMP – Simple Network Management Protocol) é uma estrutura para gerenciar dispositivos em uma rede usando a pilha de protocolos TCP/IP. Um gerente, normalmente uma estação, controla e monitora um conjunto de agentes, geralmente roteadores. O SNMP utiliza os serviços da SMI e da MIB. A SMI nomeia objetos, define os tipos de dados que podem ser armazenados em um objeto e codifica os dados. A MIB é uma coleção de grupos de objetos que podem ser gerenciados pelo SNMP. A MIB usa ordenação lexicográfica para gerenciar suas variáveis.

A Notação Sintática Abstrata Um (ASN.1 – Abstract Syntax Notation Number One) é uma linguagem que define a sintaxe e a semântica de dados. Ela usa alguns símbolos, palavras-chave e tipos de dados simples e estruturados. Parte da ASN.1 é usada pela SMI para definir o formato de objetos e os valores utilizados no gerenciamento da rede.

9.5 ATIVIDADES PRÁTICAS

9.5.1 Testes

Diversos testes interativos relativos a este capítulo podem ser encontrados no *site* www.mhhe.com/forouzan. Recomendamos que o estudante realize os testes para verificar o seu entendimento do material apresentado antes de continuar com as atividades práticas.

Questões

9-1 Qual das seguintes opções não corresponde a uma das cinco áreas de gerenciamento de redes definidas pela ISO?

a. falhas **b.** desempenho **c.** pessoas

9-2 Qual das seguintes opções não faz parte do gerenciamento de configuração?

a. reconfiguração
b. criptografia
c. documentação

9-3 Um gerente de rede decide substituir o antigo roteador que conecta a organização à Internet por outro mais potente. Qual área de gerenciamento de redes está envolvida aqui?

9-4 Um gerente de rede decide substituir uma versão do *software* de contabilização por uma nova versão. Qual área de gerenciamento de redes está envolvida aqui?

9-5 Quais são as diferenças e semelhanças entre o gerenciamento de falhas reativo e o gerenciamento de falhas proativo?

9-6 Se o gerente de rede não substitui um componente cuja vida útil tenha sido expirada, qual área do gerenciamento de redes foi ignorada?

9-7 Quais são as diferenças e semelhanças entre o tráfego de dados interno e externo em uma organização?

9-8 Se um estudante em uma faculdade é capaz de monopolizar o acesso a certo *software*, fazendo com que os outros alunos esperem um longo tempo para poder utilizá-lo, qual área de gerenciamento de redes falhou?

9-9 Qual dos seguintes dispositivos não pode ser uma estação gerente no SNMP?

a. um roteador
b. uma estação
c. um *switch*

9-10 O gerente SNMP executa um programa-cliente SNMP ou um programa-servidor SNMP?

9-11 Mostre como o nome textual "iso.org.dod" é numericamente codificado usando a SMI.

9-12 É possível existir um nome textual em SMI da forma "iso.org.internet"? Explique.

9-13 Determine o tipo (simples, sequência, sequência de) dos seguintes objetos em SMI.

a. Um inteiro sem sinal
b. Um endereço IP
c. Um nome de objeto
d. Uma lista de inteiros
e. Um registro definindo um nome de objeto, um endereço IP e um número inteiro
f. Uma lista de registros na qual cada registro é um nome de objeto seguido por um contador

9-14 Qual é o comprimento do campo de valor na codificação BER a seguir?

04	09	48	65	6C	4C	...

9-15 Quais são as diferenças e semelhanças entre a SMI e a MIB?

9-16 Na MIB, o que o objeto *if* define? Por que esse objeto precisa ser gerenciado?

9-17 Considere que um identificador de objeto na MIB possua três variáveis simples. Se o identificador do objeto é *x*, qual é o identificador de cada variável?

9-18 O SNMP é capaz de referenciar a linha toda de uma tabela? Em outras palavras, o SNMP é capaz de obter ou modificar os valores de uma linha toda de uma tabela? Explique.

9-19 Uma mensagem SNMP é capaz de referenciar um nó folha da árvore MIB? Explique.

9-20 Considere que um objeto gerenciável tenha apenas três variáveis simples. Quantas folhas podem ser encontradas na árvore MIB dele?

9-21 Suponha que um objeto gerenciável tenha uma tabela com três colunas. Quantas folhas existem na árvore MIB dessa tabela?

9-22 Quais são as diferenças e semelhanças entre uma PDU GetRequest e uma PDU SetRequest?

9-23 No SNMP, qual das seguintes PDUs é enviada por um cliente SNMP para um servidor SNMP?

a. GetRequest
b. Response
c. Trap

9-24 Quais são os números de porta de origem e destino quando uma mensagem SNMP transporta cada uma das seguintes PDUs?

a. GetRequest
b. Response
c. Trap
d. Report

Problemas

9-1 Suponha que o objeto *x* apresente duas variáveis simples: um inteiro e um endereço IP. Qual é o identificador para cada variável?

9-2 Suponha que o objeto *x* apresente apenas uma tabela com duas colunas. Qual é o identificador para cada coluna?

9-3 Suponha que o objeto *x* apresente duas variáveis simples e uma tabela com duas colunas. Qual é o identificador para cada variável e cada coluna da tabela? Assuma que as variáveis simples venham antes da tabela.

9-4 Suponha que o objeto *x* tenha uma variável simples e duas tabelas com duas e três colunas, respectivamente. Qual é o identificador da variável e de cada coluna de cada tabela? Assuma que a variável simples venha antes das tabelas.

9-5 O objeto *x* tem duas variáveis simples. Como o SNMP pode fazer referência à instância de cada variável?

9-6 O objeto *x* tem uma tabela com duas colunas. A tabela, neste momento, tem três linhas com os conteúdos mostrados a seguir. Se o índice da tabela é baseado nos valores da primeira coluna, mostre como o SNMP pode acessar cada instância.

Objeto *x*

a	aa
b	bb
c	cc

Tabela

9-7 Um dos objetos (grupos) que podem ser gerenciados é o grupo ***ip*** com o identificador de objeto (1.3.6.1.2.1.4) no qual (1.3.6.1.2.1) é o identificador da mib-2 e (4) define o grupo ***ip***. Em um agente, esse objeto tem 20 variáveis simples e três tabelas. Uma das tabelas é a tabela de roteamento (encaminhamento) com o identificador (1.3.6.1.2.1.4.21). Essa tabela tem onze colunas, a primeira das quais é chamada de *ipRouteDes*, que significa o endereço IP de destino. Considere que a indexação seja baseada na primeira coluna, e também que a tabela tenha quatro linhas no momento, com os endereços IP de destino (201.14.67.0), (123.16.0.0), (11.0.0.0) e (0.0.0.0). Mostre como o SNMP pode acessar todas as quatro instâncias da segunda coluna, denominada *ipRouteIfIndex*, a qual define os números de interface pelos quais o pacote IP deve ser enviado.

9-8 Mostre a codificação do inteiro (INTEGER) 1456 usando BER.

9-9 Mostre a codificação da sequência de caracteres (OCTET STRING) "Ola mundo" usando BER.

9-10 Mostre a codificação do endereço IP (IPAddress) 112.56.23.78 usando BER.

9-11 Mostre a codificação do identificador de objeto 1.3.6.1.2.1.7.1 (a variável udpInDatagram no grupo udp) usando BER.

9-12 Usando BER, mostre como podemos codificar um tipo de dados estruturado composto por um inteiro (INTEGER) de valor (2371), uma sequência de caracteres (OCTET STRING) de valor "Computador", e um endereço IP (IPAddress) de valor (185.32.1.5), conforme mostrado a seguir:

```
SEQUENCE
{
    INTEGER 2371
    OCTET STRING "Computador"
    IPAddress 185.32.1.5
}
```

9-13 Considere que uma estrutura de dados seja formada por um número inteiro (INTEGER) de valor 131 e que outra estrutura seja formada por um endereço IP (IPAddress) de valor 24.70.6.14 e por uma sequência de caracteres (OCTET STRING) de valor "UDP". Usando BER, mostre a codificação dessa estrutura de dados.

9-14 Dado o código **02040000C738**, decodifique-o usando BER.

9-15 Dado o código **300C020400000998060 40A05030E**, decodifique-o usando BER.

9-16 Dado o código **300D04024E6F3007060 30103060500**, decodifique-o usando BER.

9-17 Considere que um gerente precise saber o número de datagramas de usuário que um agente enviou (um contador udpOutDatagrams com o identificador 1.3.6.1.2.1.7.4). Mostre o código para um par variável-valor (Varbind) que é enviado em uma mensagem do tipo GetRequest e o código que o agente enviará na mensagem de resposta (Response) se o valor do contador naquele momento for 15.

9-18 Defina uma mensagem SNMP (ver Figura 9.19) utilizando a sintaxe definida para tipos de dados estruturados no ASN.1.

9-19 Defina uma PDU do tipo GetRequest (ver Figura 9.18) utilizando a sintaxe definida para tipos de dados estruturados do ASN.1.

9-20 Defina uma PDU do tipo Response (ver Figura 9.18) utilizando a sintaxe definida para tipos de dados estruturados do ASN.1.

9-21 Defina uma lista de pares variável-valor (Varbind) (ver Figura 9.19) utilizando a sintaxe definida para tipos de dados estruturados do ASN.1.

CAPÍTULO

10 SEGURANÇA DE REDES

O tópico de segurança de redes é muito amplo e envolve algumas áreas específicas da matemática, como a teoria dos números, por exemplo. Neste capítulo, buscamos dar uma introdução bastante simples sobre esse tema com o objetivo de preparar o terreno para estudos mais aprofundados. Dividimos este capítulo em cinco seções.

- Na primeira seção, introduzimos os conceitos de segurança de redes. Discutimos metas de segurança, tipos de ataques e os serviços fornecidos pelos mecanismos de segurança de redes. Mencionamos duas técnicas de segurança: criptografia e esteganografia.
- Na segunda seção, apresentamos a primeira meta da segurança, a confidencialidade. Discutimos cifras de chave simétrica e cifras de chave assimétrica, bem como suas aplicações. Mostramos que cifras de chave simétrica podem ser usadas para mensagens longas, enquanto cifras de chave assimétrica podem ser usadas basicamente para mensagens curtas. Precisamos de ambas atualmente.
- Na terceira seção, discutimos outros aspectos da segurança: integridade e autenticação de mensagens, assinatura digital, autenticação de entidades e gerenciamento de chaves. Esses aspectos, hoje em dia, fazem parte de sistemas de segurança de forma complementar à confidencialidade. Veremos que, algumas vezes, a confidencialidade não pode ser obtida a menos que adicionemos um ou mais desses aspectos.
- Na quarta seção, aplicamos o que aprendemos nas três primeiras seções no cenário da Internet. A segurança pode ser aplicada na camada de aplicação, na de transporte ou na de rede. Primeiramente, discutimos a segurança na camada de aplicação: descrevemos dois protocolos de segurança usados para proteger mensagens de correio eletrônico (e-mail): Privacidade Bastante Boa (PGP – Pretty Good Privacy) e o S/MIME. Discutimos, então, mecanismos de segurança na camada de transporte que podem proteger qualquer protocolo da camada de aplicação: o SSL e o TLS. Finalmente, discutimos a segurança na camada de rede com o IPSec, que é usado para fornecer segurança para aplicações que utilizam a camada de transporte ou para aquelas que utilizam diretamente os serviços da camada de rede.
- Na quinta seção, discutimos *firewalls*. Mostramos como podemos prevenir que mensagens prejudiciais alcancem um sistema. Discutimos primeiramente os *firewalls* de filtragem de pacotes, que operam nas camadas de rede e transporte. Em seguida, discutimos *firewalls proxy*, que operam na camada de aplicação.

10.1 INTRODUÇÃO

Vivemos na era da informação. Precisamos manter informações sobre cada aspecto de nossas vidas. Em outras palavras, a informação é um ativo que tem valor, assim como qualquer outro. Como um

ativo, e para ser considerada segura, a informação precisa ser protegida contra ataques, acessos não autorizados (*confidencialidade*) e alterações não autorizadas (*integridade*); também deve estar disponível para uma entidade autorizada quando necessário (*disponibilidade*).

Nas últimas três décadas, redes de computadores criaram uma revolução na utilização de informações, que agora são distribuídas. Pessoas autorizadas podem enviar e receber informações à distância usando redes de computadores. Embora os três requisitos mencionados anteriormente – confidencialidade, integridade e disponibilidade – não tenham mudado, agora eles assumem algumas novas dimensões. As informações não precisam ser confidenciais apenas quando armazenadas; também deve haver uma maneira de manter a sua confidencialidade enquanto são transmitidas de um computador para outro.

Nesta seção, discutimos primeiramente as três metas principais da segurança da informação. Em seguida, veremos como ataques podem ameaçar essas três metas. Discutimos, então, os serviços de segurança relacionados a elas. Finalmente, descrevemos duas técnicas que implementam as metas de segurança e previnem ataques.

10.1.1 Metas de segurança

Discutiremos, primeiramente, três metas de segurança: confidencialidade, integridade e disponibilidade.

Confidencialidade

A *confidencialidade* é provavelmente o aspecto mais comum da segurança da informação. Precisamos proteger nossas informações confidenciais. Uma organização precisa se proteger contra ações maliciosas que comprometem a confidencialidade de suas informações. A confidencialidade não se aplica apenas ao armazenamento, mas também à transmissão de informações. Quando enviamos uma informação para que ela seja armazenada em um computador remoto, ou quando obtemos uma informação vinda de um computador remoto, precisamos ocultá-la durante a transmissão.

Integridade

As informações precisam ser modificadas constantemente. Em um banco, quando um cliente deposita ou retira dinheiro, o saldo de sua conta precisa ser alterado. *Integridade* significa que as mudanças devem poder ser feitas apenas por entidades autorizadas e por meio de mecanismos autorizados. Uma violação de integridade não é necessariamente o resultado de um ato malicioso; uma interrupção no sistema, como uma oscilação de energia, pode também criar alterações indesejadas em alguma informação.

Disponibilidade

O terceiro componente da segurança da informação é a disponibilidade. As informações criadas e armazenadas por uma organização precisam estar disponíveis para entidades autorizadas. A informação é inútil se ela não estiver disponível. As informações precisam ser constantemente modificadas, o que significa que devem estar acessíveis a entidades autorizadas. A indisponibilidade de informações é tão prejudicial para uma organização quanto a ausência de confidencialidade ou integridade. Imagine o que aconteceria a um banco se os clientes não pudessem acessar suas contas para realizar transações.

10.1.2 Ataques

Nossas três metas de segurança – confidencialidade, integridade e disponibilidade – podem ser ameaçadas por *ataques*. Embora a literatura utilize diferentes abordagens para categorizar ataques, podemos dividi-los em três grupos relacionados às metas de segurança. A Figura 10.1 mostra essa taxonomia.

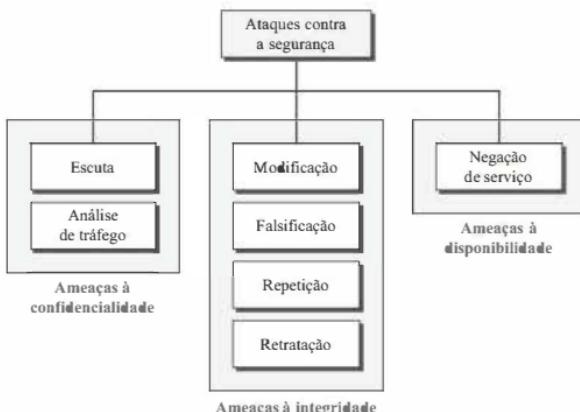



Figura 10.1 Taxonomia de ataques relacionados às metas de segurança.

Ataques ameaçando a confidencialidade

Em geral, dois tipos de ataques ameaçam a confidencialidade das informações: *escuta* e *análise de tráfego*.

Escuta

A escuta refere-se ao acesso não autorizado ou à intercepção de dados. Por exemplo, um arquivo transferido usando a Internet pode conter informações confidenciais. Uma entidade não autorizada pode interceptar a transmissão e usar o conteúdo em seu benefício próprio. Para evitar escutas, pode-se fazer com que os dados sejam incompreensíveis pelo interceptador usando técnicas de cifração, que serão discutidas mais adiante.

Análise de tráfego

Embora a cifração seja capaz de tornar os dados incompreensíveis pelo interceptador, o atacante pode obter algum outro tipo de informação monitorando o tráfego na rede. Por exemplo, ele pode descobrir o endereço eletrônico (como o endereço de e-mail) do emissor ou do receptor. Pode coletar pares de solicitações e respostas que o ajudem a adivinhar a natureza da transação.

Ataques ameaçando a integridade

A integridade dos dados pode ser ameaçada por diversos tipos de ataques: *modificação*, *falsificação*, *repetição* e *retratação*.

Modificação

Depois de interceptar ou acessar uma informação, o atacante modifica essa informação de modo que ela o beneficie. Por exemplo, um cliente envia uma mensagem para um banco para iniciar alguma transação. O atacante intercepta a mensagem e modifica o tipo de transação de modo a beneficiá-lo. Perceba que algumas vezes o atacante simplesmente destrói ou atrasa a mensagem para prejudicar o sistema ou para tirar algum proveito desse fato.

Falsificação

A falsificação acontece quando o atacante se faz passar por outra pessoa. Por exemplo, um atacante pode roubar o cartão do banco e a senha de um cliente do banco e fingir ser aquele cliente. Algumas vezes, o atacante pode fingir ser a entidade que é o destinatário de uma mensagem. Por exemplo, um usuário tenta entrar em contato com um banco, mas alguém (outro *site*) finge ser o banco e obtém alguma informação do usuário.

Repetição

A repetição é outro tipo de ataque. O atacante obtém uma cópia de uma mensagem enviada por um usuário e mais tarde tenta enviá-la novamente. Por exemplo, uma pessoa envia uma solicitação ao seu banco requisitando que um pagamento seja efetuado para o atacante, que prestou um serviço para ela. O atacante intercepta a mensagem e a envia novamente para receber um novo pagamento do banco.

Retratação

Esse tipo de ataque é diferente dos outros porque é realizado por uma das duas partes envolvidas na comunicação: o emissor ou o receptor. O emissor da mensagem pode, posteriormente, negar ter enviado a mensagem, ou o receptor pode negar ter recebido a mensagem. Um exemplo de retratação pelo emissor seria um cliente pedindo a seu banco que envie dinheiro a um terceiro e posteriormente negar que tenha feito tal pedido. Um exemplo de retratação por parte do receptor pode ocorrer quando uma pessoa compra um produto de um fabricante e paga por ele eletronicamente, mas depois o fabricante nega ter recebido o pagamento e exige ser pago.

Ataques ameaçando a disponibilidade

Mencionamos apenas um ataque capaz de ameaçar a disponibilidade: a *negação de serviço*.

Negação de serviço

A negação de serviço (DoS* – Denial of Service) é um ataque muito comum. Ele pode reduzir a velocidade ou interromper completamente o serviço fornecido por um sistema. O atacante pode usar várias estratégias para atingir esse objetivo. Pode enviar uma quantidade tão grande de solicitações falsas para um servidor que este acaba travando devido à elevada carga de computação. O atacante pode interceptar e destruir a resposta enviada pelo servidor para um cliente, levando o cliente a acreditar que o servidor não está respondendo. Também pode interceptar as solicitações dos clientes, fazendo com que eles enviem pedidos diversas vezes e sobrecarreguem o sistema.

10.1.3 Serviços e técnicas

A ITU-T define alguns serviços para atingir metas de segurança e prevenir ataques. Cada um desses serviços é projetado para evitar um ou mais ataques, preservando as metas de segurança. A implementação efetiva das metas de segurança requer algumas técnicas, duas delas predominantes atualmente: uma é bastante geral (criptografia) e a outra é mais específica (esteganografia).

Criptografia

Alguns serviços de segurança podem ser implementados usando criptografia. A **criptografia**, uma palavra de origem grega, significa "escrita secreta". No entanto, usamos o termo quando nos referimos à ciência e à arte de transformar mensagens com o objetivo de torná-las seguras e imunes a

* N. de T.: Uma variação muito comum do DoS é o DDoS (*Distributed Denial of Service*, ou ataque de negação de serviço distribuído) no qual o ataque de negação é realizado por diversas entidades (atacantes) simultaneamente.

ataques. Embora, no passado, o termo *criptografia* se referisse apenas à **cifração** e à **decifração** de mensagens usando chaves secretas, atualmente sua definição envolve três mecanismos distintos: cifração por chave simétrica, cifração por chave assimétrica e geração de *hash*. Discutiremos todos esses mecanismos mais adiante.

Esteganografia

Embora este capítulo e o próximo sejam, basicamente, a respeito da criptografia como técnica para criar mecanismos de segurança, há outra técnica bastante usada para comunicações secretas no passado e que está resurgindo nos tempos atuais: a esteganografia. A palavra **esteganografia**, de origem grega, significa "escrita oculta", diferentemente da criptografia, que significa "escrita secreta". *Criptografia* significa esconder o conteúdo de uma mensagem por meio de cifração; já *esteganografia* significa ocultar a mensagem em si, encobrindo-a com alguma outra coisa. Deixamos a discussão sobre esteganografia para os livros dedicados a esse tema.

10.2 CONFIDENCIALIDADE

Agora, voltemos nossa atenção para a primeira meta de segurança, a confidencialidade. A confidencialidade pode ser obtida usando cifras, que podem ser separadas em duas grandes categorias: de chave simétrica e de chave assimétrica.

10.2.1 Cifras de chave simétrica

Uma **cifra de chave simétrica** (ou simplesmente **cifra simétrica**) utiliza a mesma chave tanto na cifração como na decifração e a chave pode ser usada para comunicação bidirecional. É por isso que a cifra é denominada *simétrica*. A Figura 10.2 mostra a ideia geral por trás de uma cifra de chave simétrica.

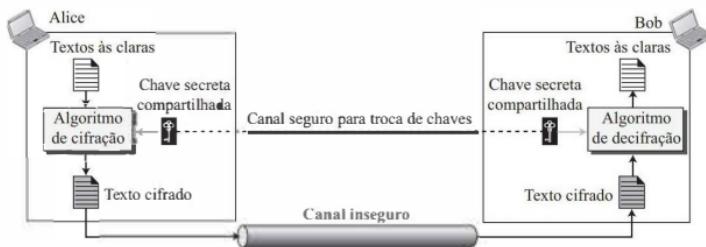



Figura 10.2 Ideia geral de uma cifra de chave simétrica.

Cifras de chave simétrica também são denominadas cifras de chave secreta.

Na Figura 10.2, uma entidade, Alice, pode enviar uma mensagem para outra entidade, Bob, por meio de um canal inseguro considerando que um atacante, Eva, não seja capaz de compreender o conteúdo da mensagem simplesmente escutando o canal.

A mensagem original enviada por Alice para Bob é chamada **texto às claras** (*plaintext*); a mensagem enviada pelo canal é chamada **texto cifrado** (*ciphertext*). Para criar o texto cifrado a partir do texto às claras, Alice usa um **algoritmo de cifração** e uma *chave secreta compartilhada*.

Para recuperar o texto às claras a partir do texto cifrado, Bob usa um **algoritmo de decifração** e a mesma chave secreta. Os algoritmos de cifração e decifração são denominados **cifras**. Uma *chave* é um conjunto de valores (números) sobre o qual a cifra, como um algoritmo, opera.

Perceba que a cifração por chave simétrica utiliza uma única chave (a chave em si pode ser um conjunto de valores) tanto para a cifração quanto para a decifração. Além disso, os algoritmos de cifração e decifração são inversos um do outro. Se P é o texto às claras, C é o texto cifrado e K é a chave, o algoritmo de cifração $E_k(x)$ cria o texto cifrado a partir do texto às claras; o algoritmo de decifração $D_k(x)$ cria o texto original a partir do texto cifrado. Consideramos que $E_k(x)$ e $D_k(x)$ sejam inversos um do outro: eles cancelam o efeito um do outro se forem aplicados sequencialmente sobre a mesma entrada. Temos

Cifração: $C = E_k(P)$ | **Decifração: $P = D_k(C)$**

onde $D_k(E_k(x)) = E_k(D_k(x)) = x$. É preciso enfatizar que é melhor tornar os algoritmos de cifração e de decifração públicos, mas manter a chave compartilhada em segredo. Isto significa que Alice e Bob precisam de outro canal, seguro, para trocar a chave secreta. Alice e Bob podem se encontrar em algum momento e trocar a chave pessoalmente. O canal seguro neste caso é o acordo da chave face a face. Eles também podem confiar em um terceiro para entregar-lhes a mesma chave. Ou então eles podem criar uma chave secreta temporária utilizando outro tipo de cifra – cifras de chave assimétrica – que será descrita mais adiante.

A cifração pode ser imaginada como o ato de trancar a mensagem em uma caixa; a decifração pode ser imaginada como o ato de destrancar a caixa. Na cifração por chave simétrica, a mesma chave é usada para trancar e destrancar, conforme mostra a Figura 10.3. Algumas seções mais adiante mostram que a cifração por chave assimétrica precisa de duas chaves, uma para trancar e outra para destrancar a caixa.

Figura 10.3 Cifração por chave simétrica como o ato de trancar e destrancar com a mesma chave.

As cifras de chave simétrica podem ser divididas em cifras tradicionais e cifras modernas. As cifras tradicionais são simples, orientadas a caracteres, e não são consideradas seguras com base nos padrões atuais. Cifras modernas, por outro lado, são complexas, orientadas a *bits*, e são mais seguras. Discutimos brevemente as cifras tradicionais com o objetivo de fornecer a base necessária para discutir as cifras modernas, mais complexas.

Cifras de chave simétrica tradicionais

Cifras tradicionais pertencem ao passado. Entretanto, discutiremos brevemente essas cifras aqui porque elas podem ser entendidas como os componentes das cifras modernas. Para ser mais exato, podemos dividir cifras tradicionais em cifras de substituição e cifras de transposição.

Cifras de substituição

Uma **cifra de substituição** troca um símbolo por outro. Se os símbolos no texto às claras são caracteres do alfabeto, podemos substituir um caractere por outro. Por exemplo, podemos substituir a letra A pela letra D e a letra T pela letra Z. Se os símbolos forem dígitos (0 a 9), podemos substituir 3 por 7 e 2 por 6.

Uma cifra de substituição substitui um símbolo por outro.

Cifras de substituição podem ser categorizadas como cifras monoalfabéticas ou polialfabéticas.

Cifras monoalfabéticas Neste tipo de cifra, um caractere (ou um símbolo) do texto às claras é sempre substituído pelo mesmo caractere (ou símbolo) no texto cifrado, independentemente de sua posição no texto. Por exemplo, se o algoritmo diz que a letra A no texto às claras é transformada na letra D, toda letra A é substituída pela letra D. Em outras palavras, a relação entre as letras no texto às claras e no texto cifrado é um para um.

A cifra monoalfabética mais simples é a **cifra aditiva** (ou **cifra de deslocamento**). Considere que o texto original seja composto por letras minúsculas (a até z), e que o texto cifrado seja composto por letras maiúsculas (A até Z). Para ser capaz de aplicar operações matemáticas ao texto às claras e cifrado, atribuímos valores numéricos para cada letra (minúscula ou maiúscula), conforme mostra a Figura 10.4.

Texto às claras →	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Texto cifrado →	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Valor →	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Figura 10.4 Representação dos caracteres no texto às claras e cifrado em módulo 26.

Na Figura 10.4, a cada caractere (minúsculo ou maiúsculo) é atribuído um número inteiro módulo 26. A chave secreta entre Alice e Bob também é um número inteiro módulo 26. O algoritmo de cifração soma a chave ao caractere do texto às claras; já o algoritmo de decifração subtrai a chave do caractere no texto cifrado. Todas as operações são feitas usando aritmética módulo 26.

Em uma cifra aditiva, o texto às claras, o texto cifrado e a chave são inteiros módulo 26.

Historicamente, as cifras aditivas são também chamadas de cifras de deslocamento, porque o algoritmo de cifração pode ser interpretado como "desloque *chave* caracteres para baixo" e o algoritmo de decifração pode ser interpretado como "desloque *chave* caracteres para cima". O imperador romano Júlio César usava uma cifra aditiva com um valor de chave igual a 3 para se comunicar com seus oficiais. Por isso, cifras aditivas são também chamadas **cifra de César**.

Exemplo 10.1

Use a cifra aditiva com chave = 15 para cifrar a mensagem "hello".

Solução

Aplicamos o algoritmo de cifração ao texto original, caractere a caractere:

Texto às claras: h → 07	Cifração: (07 + 15) mod 26	Texto cifrado: 22 → W
Texto às claras: e → 04	Cifração: (04 + 15) mod 26	Texto cifrado: 19 → T
Texto às claras: l → 11	Cifração: (11 + 15) mod 26	Texto cifrado: 00 → A
Texto às claras: l → 11	Cifração: (11 + 15) mod 26	Texto cifrado: 00 → A
Texto às claras: o → 14	Cifração: (14 + 15) mod 26	Texto cifrado: 03 → D

O resultado é "WTAAD". Perceba que a cifra é monoalfabética porque duas aparições do mesmo caractere no texto às claras (*l*) são cifradas como o mesmo caractere (*A*).

Exemplo 10.2

Use a cifra aditiva com chave = 15 para decifrar a mensagem "WTAAD".

Solução

Aplicamos o algoritmo de decifração ao texto cifrado, caractere a caractere:

Texto às claras: W → 22	Cifração: (22 − 15) mod 26	Texto cifrado: 07 → h
Texto às claras: T → 19	Cifração: (19 − 15) mod 26	Texto cifrado: 04 → e
Texto às claras: A → 00	Cifração: (00 − 15) mod 26	Texto cifrado: 11 → l
Texto às claras: A → 00	Cifração: (00 − 15) mod 26	Texto cifrado: 11 → l
Texto às claras: D → 03	Cifração: (03 − 15) mod 26	Texto cifrado: 14 → o

O resultado é "hello". Perceba que a operação é feita usando aritmética módulo 26, o que significa que precisamos somar 26 a um resultado negativo (por exemplo, −15 toma-se 11).

Cifras aditivas são vulneráveis a ataques usando buscas exaustivas de chaves (ataques de força bruta). O domínio de chaves da cifra aditiva é muito pequeno; existem apenas 26 chaves. Entretanto, uma das chaves, zero, é inútil (o texto cifrado resultante é idêntico ao texto às claras). Sobram apenas 25 chaves possíveis. Eva* pode facilmente iniciar um ataque de força bruta contra o texto cifrado. Uma solução melhor é criar um mapeamento entre cada caractere do texto às claras e o caractere correspondente no texto cifrado. Alice e Bob podem concordar em usar uma tabela que mostre o mapeamento para cada caractere. A Figura 10.5 mostra um exemplo de tal mapeamento.

Texto às claras →	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Texto cifrado →	N	O	A	T	R	B	E	C	F	U	X	D	Q	G	Y	L	K	H	V	I	J	M	P	Z	S	W

Figura 10.5 Uma chave exemplo para uma cifra de substituição monoalfabética.

Exemplo 10.3

Podemos usar a chave da Figura 10.5 para cifrar a seguinte mensagem**

Texto às claras:	this message is easy to encrypt but hard to find the key
Texto cifrado:	ICFVQRVVNERFVRNVSIYRGAHSLIOJICNHTIYBFGTICRXRS

Cifras polialfabéticas Neste tipo de cifra cada ocorrência de um caractere pode ter um substituto diferente. A relação entre um caractere no texto original e um caractere no texto cifrado é um para muitos. Por exemplo, a letra "a" poderia ser cifrada como "D" no início do texto, porém como "N" no meio. Cifras polialfabéticas têm como vantagem a capacidade de esconder a frequência de letras do idioma subjacente. Eva não pode usar a frequência estatística de letras distintas para quebrar o texto cifrado.

* N. de T.: Eva é um nome fictício normalmente atribuído para identificar o atacante nas explicações de cifras. Outro nome fictício para identificar um atacante é Charlie.

** N. de T.: O texto às claras, em inglês, significa "Esta mensagem pode ser facilmente cifrada mas é difícil encontrar sua chave".

Para criar uma cifra polialfabética, precisamos que cada caractere cifrado dependa tanto do caractere correspondente no texto às claras quanto da posição daquele caractere na mensagem. Isto implica que nossa chave deve ser uma sequência (ou fluxo) de subchaves, de modo que cada subchave dependa de alguma forma da posição do caractere no texto às claras que usa aquela subchave em sua cifração. Ou seja, precisamos ter uma sequência de chaves $k = (k_1, k_2, k_3, ...)$, na qual k_i é usada para cifrar o i-ésimo caractere do texto às claras, gerando o i-ésimo caractere do texto cifrado.

Para entender como a chave depende da posição, discutiremos uma cifra polialfabética simples chamada **cifra de autochave**. Nela, a chave é uma sequência de subchaves, de modo que cada subchave é usada para cifrar o caractere correspondente no texto original. A primeira subchave é um valor predeterminado secretamente, acordado entre Alice e Bob. A segunda subchave é o valor do primeiro caractere do texto às claras (entre 0 e 25). A terceira subchave é o valor do segundo caractere do texto às claras, e assim por diante.

$P = P_1P_2P_3\ldots$ $\quad C = C_1C_2C_3\ldots$ $\quad k = (k_1, P_1, P_2, \ldots)$

Cifração: $C_i = (P_i + k_i) \bmod 26$ $\quad$ Decifração: $P_i = (C_i - k_i) \bmod 26$

O nome da cifra, *autochave*, indica que as subchaves são criadas automaticamente a partir dos caracteres do texto às claras, durante o processo de cifração.

Exemplo 10.4

Considere que Alice e Bob concordaram em usar uma cifra de autochave com o valor inicial da chave $k_1 = 12$. Agora, Alice quer enviar a Bob a mensagem "Attack is today"*. A cifração é realizada caractere a caractere. Cada caractere do texto às claras é primeiramente substituído pelo seu valor inteiro. A primeira subchave é somada para criar o primeiro caractere do texto cifrado. O restante da chave é criado à medida que os caracteres do texto às claras são lidos. Perceba que a cifra é polialfabética porque as três ocorrências da letra "a" no texto original são cifradas de forma diferente. As três ocorrências de "t" também são cifradas de forma distinta.

Texto às claras:	a	t	t	a	c	k	i	s	t	o	d	a	y
Valores de P:	00	19	19	00	02	10	08	18	19	14	03	00	24
Sequência de chaves:	*12*	*00*	*19*	*19*	*00*	*02*	*10*	*08*	*18*	*19*	*14*	*03*	*00*
Valores de C:	12	19	12	19	02	12	18	00	11	07	17	03	24
Texto cifrado:	**M**	**T**	**M**	**T**	**C**	**M**	**S**	**A**	**L**	**H**	**R**	**D**	**Y**

Cifras de transposição

Uma **cifra de transposição** não substitui um símbolo por outro; na verdade, ela modifica a localização dos símbolos. Um símbolo na primeira posição do texto às claras pode aparecer na décima posição do texto cifrado. Um símbolo na oitava posição do texto às claras pode aparecer na primeira posição do texto cifrado. Em outras palavras, um cifra de transposição reordena (transpõe) os símbolos.

Uma cifra de transposição reordena símbolos.

Considere que Alice queira enviar secretamente a mensagem "Enemy attacks tonight"** para Bob. A cifração e a decifração são mostradas na Figura 10.6. Observe que adicionamos um caractere extra (z) ao final da mensagem para fazer com que o número de caracteres seja um múltiplo de 5.

* N. de T.: "Attack is today" pode ser traduzido do inglês como "O ataque é hoje".

** N. de T.: "Enemy attacks tonight" pode ser traduzido do inglês como " O inimigo ataca hoje à noite ".

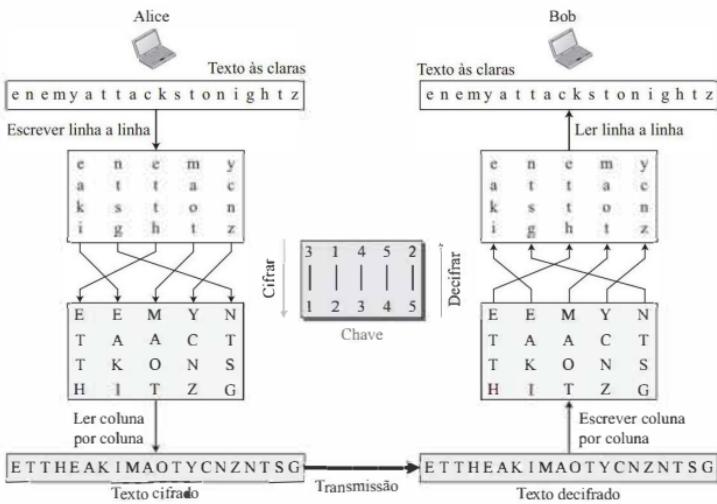



Figura 10.6 Cifra de transposição.

A primeira tabela é criada por Alice, que escreve o texto às claras linha a linha. As colunas são permutadas usando uma chave. O texto cifrado é criado lendo-se a segunda tabela coluna por coluna. Bob executa as mesmas três operações na ordem inversa. Ele escreve o texto cifrado coluna a coluna na primeira tabela, permuta as colunas, e então lê a segunda tabela linha a linha. Perceba que a mesma chave é usada para a cifração e para a decifração, mas o algoritmo aplica a chave na ordem inversa.

Cifras de bloco e de fluxo

A literatura divide as cifras simétricas em duas grandes categorias: cifras de fluxo e cifras de bloco.

Cifra de fluxo Neste tipo de cifra, a cifração e a decifração são feitas um símbolo (por exemplo, um caractere ou um *bit*) de cada vez. Temos um fluxo (sequência de símbolos) de texto às claras, um fluxo de texto cifrado, e um fluxo de chave. Denotamos o fluxo de texto às claras por P, o fluxo de texto cifrado por C, e o fluxo de chave por K.

$$P = P_1P_2P_3, \ldots \qquad C = C_1C_2C_3, \ldots \qquad K = (k_1, k_2, k_3, \ldots)$$
$$C_1 = E_{k1}(P_1) \qquad C_2 = E_{k2}(P_2) \qquad C_3 = E_{k3}(P_3) \ldots$$

Cifras de bloco Neste tipo de cifra, um grupo de símbolos do texto às claras de tamanho m ($m > 1$) é cifrado de cada vez, criando um grupo de símbolos do texto cifrado de mesmo tamanho. Com base nessa definição, em uma cifra de bloco, uma única chave é utilizada para cifrar o bloco todo mesmo se a chave for composta por diversos valores. Em uma cifra de bloco, um bloco de texto cifrado depende do bloco inteiro do texto às claras.

Combinação Na prática, os blocos do texto às claras são cifrados individualmente, mas uma sequência de chaves é usada para cifrar a mensagem inteira, bloco por bloco. Em outras palavras, a

cifra é uma cifra de bloco quando observamos os blocos individuais, mas é uma cifra de fluxo quando olhamos para a mensagem toda, considerando cada bloco como uma só unidade. Cada bloco usa uma chave diferente, que pode ser gerada antes ou durante processo de cifração.

Cifras de chave simétrica modernas

As cifras de chave simétrica tradicionais que estudamos até agora são *cifras orientadas a caracteres*. Com o surgimento do computador, passamos a precisar de *cifras orientadas a bits*. A razão é que a informação a ser cifrada não envolve apenas texto; pode também envolver números, gráficos, áudio e dados de vídeo. É conveniente converter esses tipos de dados em um fluxo de *bits*, cifrar o fluxo e, então, transmitir o fluxo cifrado. Além disso, quando o texto é tratado no nível dos *bits*, cada caractere é substituído por 8 (ou 16) *bits*, o que significa que o número de símbolos torna-se 8 (ou 16) vezes maior. Misturar um número maior de símbolos aumenta a segurança. Cifras modernas podem ser tanto de bloco como de fluxo.

Cifras de bloco modernas

Uma *cifra de bloco moderna* baseada em chaves simétricas é capaz de cifrar um bloco de *n bits* de texto às claras ou decifrar um bloco de *n bits* de texto cifrado. Os algoritmos de cifração e de decifração utilizam uma chave de *k bits*. O algoritmo de decifração deve ser o inverso do algoritmo de cifração, e ambas as operações devem usar a mesma chave secreta para que Bob possa recuperar a mensagem enviada por Alice. A Figura 10.7 ilustra a ideia geral da cifração e decifração usando uma cifra de bloco moderna.

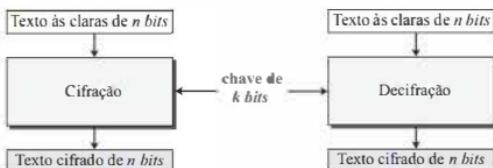



Figura 10.7 Uma cifra de bloco moderna.

Se a mensagem tiver menos do que *n bits*, devem ser adicionados *bits* de enchimento (*padding*) para completar um bloco de *n bits*; se a mensagem tiver mais do que *n bits*, ela deve ser dividida em blocos de *n bits* e o número apropriado de *bits* de enchimento deve ser adicionado ao último bloco, se necessário. Os valores geralmente encontrados para *n* são 64, 128, 256 e 512 *bits*.

Componentes de uma cifra de bloco moderna Cifras de blocos modernas são cifras de substituição quando observamos um bloco inteiro. No entanto, cifras de bloco modernas não são projetadas como uma unidade única. Para criar um algoritmo resistente a ataques, uma cifra de bloco moderna é projetada como uma combinação de unidades de transposição (algumas vezes denominadas *P-boxes*, ou *caixas-P*), unidades de substituição (algumas vezes denominadas *S-boxes*, ou *caixas-S*), operações de OU-exclusivo (XOR), elementos de deslocamento, elementos de permuta, elementos de separação e elementos de combinação. A Figura 10.8 mostra os componentes de uma cifra de bloco moderna.

Uma **caixa-P** (caixa de permutação) é o paralelo à cifra de transposição de caracteres tradicional, porém ela transpõe *bits*. Podemos encontrar três tipos de caixas-P em cifras de blocos modernas: caixas-P simples, caixas-P de expansão e caixas-P de compressão. Uma **caixa-S** (caixa de substituição, ou *S-Box*) pode ser imaginada como uma cifra de substituição em

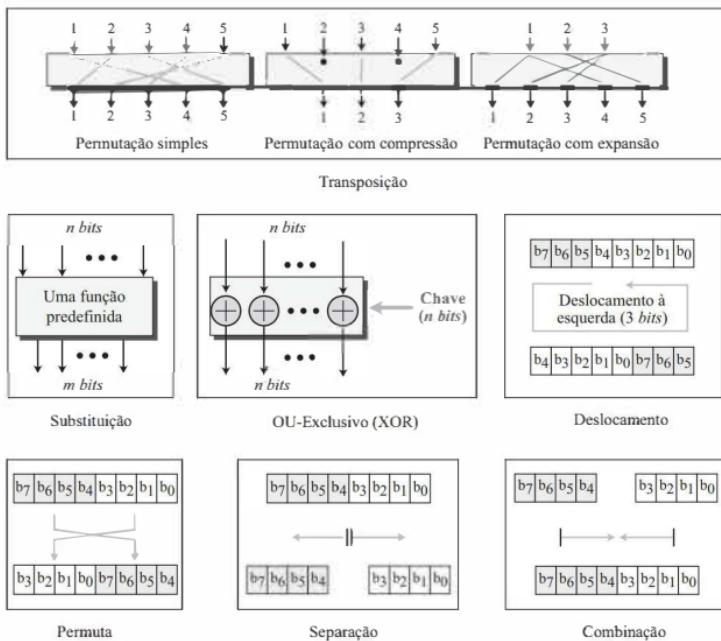



Figura 10.8 Componentes de uma cifra de bloco moderna.

miniatura, mas que substitui *bits*. Ao contrário de cifras de substituição tradicionais, uma caixa-S pode ter um número diferente de entradas e saídas. Um componente importante na maioria das cifras de bloco é a operação de *OU-exclusivo*, na qual a saída é 0 se as duas entradas forem iguais e a saída é 1 se as duas entradas forem diferentes. Em cifras de bloco modernas, usamos *n* operações de OU-exclusivo para combinar um conjunto de dados de *n bits* com uma chave de *n bits*. A operação de OU-exclusivo normalmente é a única unidade em que a chave é aplicada. Os outros componentes são normalmente baseados em funções predefinidas (sem envolvimento da chave).

Outro componente encontrado em algumas cifras de bloco modernas é a *operação de deslocamento circular*. O deslocamento pode ser para a esquerda ou para a direita. A operação de deslocamento circular para a esquerda desloca cada *bit* em uma palavra de *n bits* de *k* posições para a esquerda; os *k bits* mais à esquerda são removidos da esquerda e tornam-se os *bits* mais à direita. A *operação de permuta* é um caso especial da operação de deslocamento circular na qual o número de *bits* deslocados é $k = n/2$.

Duas outras operações encontradas em algumas cifras de blocos são a separação e a combinação. A *operação de separação* divide uma palavra de *n bits* no meio, criando duas palavras de comprimentos iguais. A *operação de combinação* normalmente concatena duas palavras de comprimentos iguais, cada uma de tamanho *n*/2 *bits*, para criar uma palavra de *n bits*.

Data Encryption Standard

Para dar um exemplo de uma cifra de bloco moderna, discutiremos o **Padrão de Cifração de Dados** (DES – Data Encryption Standard)*. A Figura 10.9 mostra os elementos do algoritmo de cifração do DES.

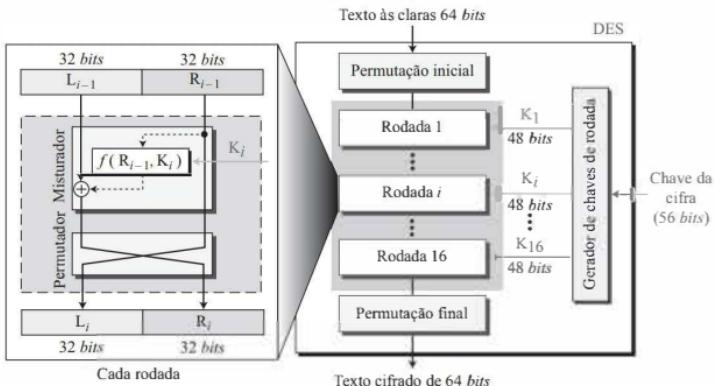



Figura 10.9 Estrutura geral do DES.

Durante a cifração, o DES processa um texto às claras de 64 *bits* e cria um texto cifrado de 64 *bits*; durante a decifração, o DES processa um texto cifrado de 64 *bits* e cria um bloco de texto às claras de 64 *bits*. A mesma chave de 56 *bits* é usada pela cifra durante a cifração e a decifração.

A permutação inicial tem como entrada 64 *bits*, que são permutados de acordo com uma regra predefinida. A permutação final é o inverso da permutação inicial. Essas duas permutações cancelam o efeito uma da outra. Ou seja, se as rodadas forem eliminadas da estrutura, o texto cifrado torna-se idêntico ao texto às claras.

Rodadas O DES opera em 16 rodadas. Cada rodada do DES é uma transformação inversível (estrutura de Feistel), conforme mostrado na Figura 10.9. A rodada recebe L_{i-1} e R_{i-1} da rodada anterior (ou da caixa de permutação inicial) e cria L_i e R_i, que vão para a próxima rodada (ou para a caixa de permutação final). Cada rodada pode ter até dois elementos da cifra (misturador e permutador). Todos esses elementos são inversíveis. O permutador é, obviamente, inversível; ele permuta a metade esquerda do texto com sua metade direita. O misturador é inversível por causa da operação de XOR. Todos os elementos não inversíveis são colocados dentro da função $f(R_{i-1}, K_i)$.

Função DES O coração do DES é a função DES. A função DES aplica uma chave de 48 *bits* aos 32 *bits* mais à direita (R_{i-1}) para produzir uma saída de 32 *bits*. Essa função é composta por quatro seções: uma caixa-P de expansão, uma função de branqueamento (que adiciona a chave da rodada), um grupo de caixas-S e uma caixa-P simples, conforme mostra a Figura 10.10.

Como R_{i-1} é uma entrada de 32 *bits* e K_i é uma chave de 48 *bits*, precisamos primeiramente expandir R_{i-1} para 48 *bits*. Essa permutação com expansão segue uma regra predeterminada.

* N. de T.: O DES é um padrão obsoleto, tendo sido substituído pelo AES (Advanced Encryption Standard, ou Padrão Avançado de Cifração) em 2001. Mais informações sobre o AES podem ser obtidas no *site* do NIST (National Institute of Standards and Technology): http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

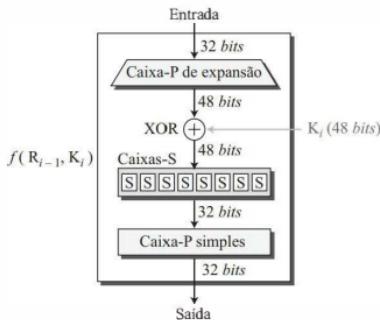



Figura 10.10 Função DES.

Após a permutação com expansão, o DES aplica a operação de XOR sobre a parte expandida da direita juntamente com a chave de rodada. As caixas-S executam a operação de combinação de fato. O DES usa oito caixas-S, cada uma com uma entrada de 6 *bits* e uma saída de 4 *bits*. A última operação na função DES é uma permutação simples com uma entrada de 32 *bits* e uma saída de 32 *bits*.

Escalonamento de chaves O gerador de chaves de rodada cria 16 chaves de 48 *bits* a partir de uma chave de 56 *bits*. No entanto, a chave da cifra é normalmente fornecida como uma chave de 64 *bits* na qual os 8 *bits* adicionais são *bits* de paridade, descartados antes do início do processo de geração de chaves (conhecido como *escalonamento de chaves*), conforme mostra a Figura 10.11.

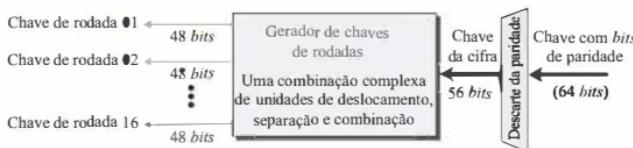



Figura 10.11 Escalonamento de chaves.

Exemplo 10.5

Escolhemos um bloco de texto às claras aleatório e uma chave aleatória e determinamos (usando um programa) qual seria o bloco de texto cifrado (em notação hexadecimal), conforme mostrado a seguir.

texto às claras:	Chave:	texto cifrado:
123456ABCD132536	AABB09182736CCDD	C0B7A8D05F3A829C

Exemplo 10.6

Para verificar a eficácia de DES quando um único *bit* é alterado na entrada, usamos dois textos às claras distintos com um único *bit* de diferença entre eles como entrada de um programa. As duas mensagens cifradas

são completamente diferentes, mesmo sem alterarmos a chave. Embora os dois blocos de texto às claras sejam diferentes apenas no *bit* mais à direita, os blocos do texto cifrado diferem em 29 *bits*.

Texto às claras:	**Chave:**	Texto cifrado:
0000000000000000	**22234512987ABB23**	**4789FD476E82A5F1**
Texto às claras:	**Chave:**	Texto cifrado:
0000000000000001	**22234512987ABB23**	**0A4ED5C15A63FEA3**

Cifras de fluxo modernas

Além de cifras de bloco modernas, também podemos usar cifras de fluxo modernas. As diferenças entre cifras de fluxo modernas e cifras de bloco modernas são similares às diferenças entre cifras de fluxo tradicionais e cifras de bloco tradicionais, que explicamos na seção anterior. Em uma *cifra de fluxo moderna*, a cifração e a decifração são realizadas *r bits* de cada vez. Temos um fluxo (sequência) de *bits* de texto às claras $P = p_n...p_2 p_1$, um fluxo de *bits* de texto cifrado $C = c_n...c_2c_1$, e um fluxo de *bits* de chave $K = k_n...k_2k_1$, onde p_i, c_i e k_i são palavras de *r bits*. A cifração é dada por $c_i = E(k_i, p_i)$, enquanto a decifração é dada por $p_i = D(k_i, c_i)$. Cifras de fluxo são mais rápidas do que cifras de bloco. A implementação em *hardware* de uma cifra de fluxo também é mais fácil. Quando precisamos cifrar fluxos binários e transmiti-los a uma taxa constante, o uso de uma cifra de fluxo é a melhor escolha. Cifras de fluxo também são mais resistentes à corrupção de *bits* durante a transmissão*.

O tipo mais simples e mais seguro de cifra de fluxo síncrona** é o chamado ***one-time pad*** (também conhecido como **cifra de uso único** ou **chave de uso único**) que foi inventado e patenteado por Gilbert Vernam. Uma cifra *one-time pad* usa um fluxo de chaves escolhido aleatoriamente para cada cifração. Tanto o algoritmo de cifração como o de decifração utilizam uma única operação de OU-exclusivo. Com base nas propriedades da operação de OU-exclusivo, os algoritmos de cifração e decifração são inversos um do outro. É importante notar que, nessa cifra, a operação de OU-exclusivo é aplicada um *bit* de cada vez. Perceba também que deve haver um canal seguro para que Alice possa enviar o fluxo de chaves para Bob (Figura 10.12).

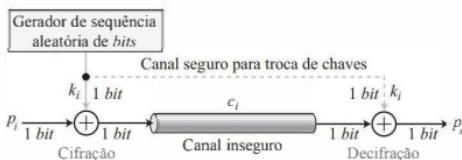



Figura 10.12 *One-time pad.*

O *one-time pad* é uma cifra ideal. Ele é perfeito. Não há uma forma pela qual um adversário seja capaz de adivinhar a chave ou estatísticas do texto às claras ou do texto cifrado. Também não existe qualquer relação entre o texto às claras e o texto cifrado. Em outras palavras, a mensagem

* N. de T.: Apesar de cifras de fluxo apresentarem vantagens sobre cifras de bloco, a experiência mostra que é mais difícil construir cifras de fluxo seguras do que cifras de bloco seguras (exceto pelo *one-time pad*). Por isso, atualmente a grande maioria das cifras padronizadas por órgãos de segurança (por exemplo, o NIST) são cifras de bloco.

** N. de T.: Em uma cifra de fluxo síncrona, a sequência (fluxo) de chaves é gerada independentemente das mensagens original e cifrada. Sua geração depende apenas de uma chave inicial K e de um valor público denominado vetor de inicialização.

cifrada é um fluxo de *bits* verdadeiramente aleatórios, mesmo se o texto às claras contiver alguns padrões. Eva não é capaz de quebrar a cifra a não ser que ela teste todas as sequências aleatórias de chave possíveis, que serão 2^n se o tamanho do texto às claras for *n bits*. Entretanto, existe um problema aqui. Como o emissor e o receptor compartilham uma chave do *one-time pad* cada vez que eles queiram se comunicar? Eles precisam de alguma forma chegar a um acordo sobre a chave aleatória. Portanto, essa cifra perfeita e ideal é muito difícil de ser utilizada, porém, existem algumas versões viáveis, menos seguras, dessa cifra. Uma das alternativas comuns é conhecida como *Registrador de Deslocamento com Realimentação* (FSR – Feedback Shift Register), mas deixamos a discussão sobre essa interessante cifra para os livros dedicados ao tema de segurança.

10.2.2 Cifras de chave assimétrica

Nas seções anteriores, discutimos as cifras de chave simétrica. Nesta seção, começamos a discussão das **cifras de chave assimétrica**. Cifras de chave simétrica e assimétrica continuarão a coexistir e a servir a comunidade de segurança. Na verdade, acreditamos que elas são complementares umas às outras; as vantagens de um tipo podem compensar as desvantagens do outro tipo.

As diferenças conceituais entre os dois sistemas são baseadas na forma como esses sistemas guardam um segredo. Na criptografia de chave simétrica, o segredo deve ser compartilhado entre duas pessoas. Na criptografia de chave assimétrica, o segredo é pessoal (não compartilhado); cada pessoa cria e mantém o seu próprio segredo.

Em uma comunidade com n pessoas, $n(n-1)/2$ segredos compartilhados são necessários na criptografia de chave simétrica; por outro lado, apenas n segredos pessoais são necessários na criptografia de chave assimétrica. Para uma comunidade cuja população total é 1 milhão, a criptografia de chave simétrica exigiria meio bilhão de segredos partilhados; já a criptografia de chave assimétrica exigiria 1 milhão de segredos pessoais.

A criptografia de chave simétrica é baseada no compartilhamento de segredos; a criptografia de chave assimétrica é baseada em segredos pessoais.

Existem alguns outros aspectos de segurança, além da cifração, que exigem o uso da criptografia de chave assimétrica, incluindo autenticação e assinaturas digitais. Sempre que uma aplicação é baseada em um segredo pessoal, precisamos usar a criptografia de chave assimétrica.

Enquanto a criptografia de chave simétrica é baseada na substituição e permutação de símbolos (caracteres ou *bits*), a criptografia de chave assimétrica é baseada na aplicação de funções matemáticas a números. Na criptografia de chave simétrica, o texto às claras e o texto cifrado são vistos como uma combinação de símbolos. A cifração e a decifração permutam esses símbolos ou substituem um símbolo por outro. Na criptografia de chave assimétrica, o texto às claras e cifrado são números; a cifração e a decifração são funções matemáticas aplicadas aos números para criar outros números.

Na criptografia de chave simétrica, os símbolos são permutados ou substituídos; na criptografia de chave assimétrica, os números são manipulados.

A criptografia de chave assimétrica utiliza duas chaves distintas: uma pública e uma privada. Se a cifração e a decifração forem vistas como o ato de trancar e destrancar cadeados com chaves, então o cadeado que é trancado com uma chave pública pode ser aberto apenas com a chave privada correspondente. A Figura 10.13 mostra que, se Alice trancar o cadeado com a chave pública de Bob, então somente a chave privada de Bob poderá abri-lo.

Figura 10.13 Trancamento e abertura em um sistema criptográfico de chave assimétrica.

A figura mostra que, ao contrário do que ocorre na criptografia de chave simétrica, existem chaves distintas na criptografia de chave assimétrica: uma **chave privada** e uma **chave pública**. Embora alguns livros usem o termo *chave secreta* em vez de *chave privada*, usamos o termo *chave secreta* apenas para a criptografia de chave simétrica e os termos *chave privada* e *chave pública* para a criptografia de chave assimétrica. Chegamos até mesmo a usar símbolos diferentes para representar as três chaves. Ou seja, queremos mostrar que uma *chave secreta* não pode ser substituída por uma *chave privada*; eles são dois tipos diferentes de segredos.

Cifras de chave assimétrica são algumas vezes denominadas cifras de chave pública.

Ideia geral

A Figura 10.14 mostra a ideia geral da criptografia de chave assimétrica quando usada para cifração.

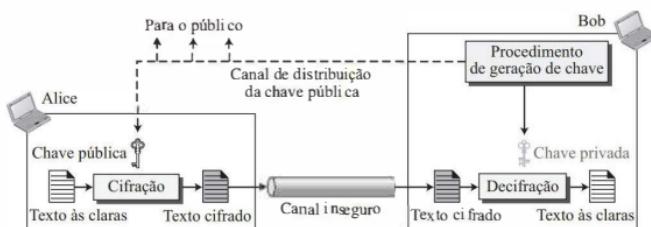



Figura 10.14 Ideia geral de um sistema criptográfico de chave assimétrica.

A Figura 10.14 mostra vários fatos importantes. Em primeiro lugar, ela enfatiza a natureza assimétrica do sistema criptográfico. O ônus de aplicar segurança fica principalmente com o receptor (Bob, neste caso). Bob precisa criar duas chaves: uma pública e uma privada. Ele é responsável por distribuir a chave pública para a comunidade, o que pode ser feito por meio de um canal de distribuição de chaves públicas. Embora esse canal não precise fornecer confidencialidade, ele deve prover autenticação e integridade. Eva não deve ser capaz de anunciar sua chave pública para a comunidade fingindo que tal chave seja a chave pública de Bob.

Em segundo lugar, usar criptografia de chave assimétrica implica que Bob e Alice não podem usar o mesmo conjunto de chaves para comunicações bidirecionais. Cada entidade na comunidade deve criar as suas próprias chaves privada e pública. A Figura 10.14 mostra como Alice pode usar a chave pública de Bob para enviar mensagens cifradas a ele. Se Bob quiser responder, Alice precisa estabelecer suas próprias chaves privada e pública.

Em terceiro lugar, o uso da criptografia de chave assimétrica implica que Bob precisa de apenas uma chave privada para receber todas as mensagens provenientes de alguém na comunidade, porém Alice precisa de *n* chaves públicas para se comunicar com *n* entidades na comunidade, uma chave pública para cada entidade. Em outras palavras, Alice precisa de um molho de chaves públicas.

Texto às claras e texto cifrado

Ao contrário da criptografia de chave simétrica, o texto às claras e o texto cifrado na criptografia de chave assimétrica são tratados como números inteiros. A mensagem deve ser codificada como um número inteiro (ou um conjunto de inteiros) antes da cifração; o número inteiro (ou o conjunto de inteiros) deve ser decodificado para gerar a mensagem após a decifração. A criptografia de chave assimétrica é normalmente usada para cifrar e decifrar informações pequenas, por exemplo, a chave criptográfica usada por uma cifra de chave simétrica. Em outras palavras, a criptografia de chave assimétrica é normalmente usada com propósitos auxiliares e não para cifrar mensagens. No entanto, esses propósitos auxiliares desempenham um papel muito importante na criptografia atual.

A criptografia de chave assimétrica é normalmente usada para cifrar ou decifrar informações pequenas.

Cifração e decifração

A cifração e a decifração na criptografia de chave assimétrica são funções matemáticas aplicadas sobre os números que representam o texto às claras e o texto cifrado. O texto cifrado pode ser visto como $C = f(K_{\text{pública}}, P)$; o texto às claras pode ser visto como $P = g(K_{\text{privada}}, C)$. A função de cifração f é usada apenas para cifrar; a função de decifração g é usada apenas para decifrar.

Necessidade de ambas

Existe um fato muito importante que algumas vezes é mal interpretado: o advento da criptografia de chave assimétrica (ou de chave pública) não elimina a necessidade da criptografia de chave simétrica (ou chave secreta). A razão é que a criptografia de chave assimétrica, que utiliza funções matemáticas para cifrar e decifrar, é muito mais lenta do que a criptografia de chave simétrica. Para cifrar mensagens grandes, a criptografia de chave simétrica ainda é necessária. Por outro lado, a velocidade da criptografia de chave simétrica não elimina a necessidade da criptografia de chave assimétrica. A criptografia de chave assimétrica ainda é necessária para prover serviços de autenticação, assinatura digital e transmissão de chaves secretas. Isto significa que, para explorarmos todos os aspectos de segurança atuais, precisamos usar tanto criptografia de chave simétrica como de chave assimétrica; uma complementa a outra.

Sistema criptográfico RSA

Embora existam vários sistemas criptográficos de chave assimétrica, um dos algoritmos de chave pública mais comum é o **sistema criptográfico RSA**, assim nomeado em função do nome de seus inventores (Rivest, Shamir e Adleman). O RSA usa dois expoentes, e e d, onde e é público e d é privado. Considere que P seja o texto às claras e que C seja o texto cifrado. Alice calcula $C = P^e \bmod n$ para criar texto cifrado C a partir do texto às claras P; Bob calcula $P = C^d \bmod n$ para recuperar o texto original enviado por Alice. O módulo n, um número muito grande, é calculado durante o processo de geração de chaves.

Procedimento

A Figura 10.15 mostra a ideia geral por trás do procedimento usado no RSA.

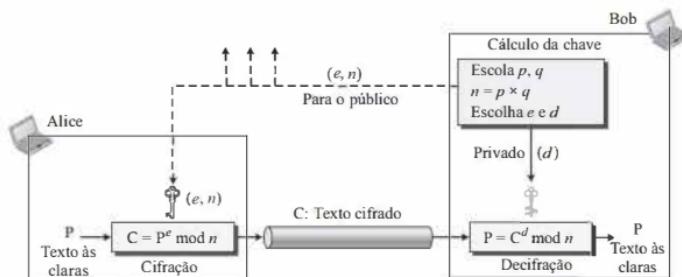



Figura 10.15 Cifração, decifração e geração de chaves no RSA.

Bob escolhe dois números primos grandes, p e q, e calcula $n = p \times q$ e $\phi = (p - 1) \times (q - 1)$. Bob, então, escolhe e e d de tal forma que $(e \times d) \bmod \phi = 1$. Bob anuncia e e n para a comunidade como a sua chave pública e mantém d como sua chave privada. Qualquer pessoa, incluindo Alice, pode cifrar uma mensagem e enviar o texto cifrado para Bob, por meio da operação $C = (P^e) \bmod n$; apenas Bob pode decifrar a mensagem por meio da operação $P = (C^d) \bmod n$. Uma atacante, Eva, não é capaz de decifrar a mensagem se p e q forem números primos muito grandes (ela não sabe o valor de d).

Exemplo 10.7

Para demonstração, suponha que Bob escolha 7 e 11 como os valores de p e q, de modo que $n = 7 \times 11 = 77$. O valor de $\phi(n)$ é calculado como $\phi(n) = (7 - 1)(11 - 1) = 60$. Se Bob escolher $e = 13$, então temos que d vale 37. Perceba que $e \times d \bmod 60 = 1$. Agora, considere que Alice queira enviar o texto às claras de valor 5 para Bob. Ela usa o expoente público 13 para cifrar o valor 5. Esse sistema não é seguro porque p e q são pequenos.

Texto às claras: 5
$C = 5^{13} = 26 \bmod 77$
Texto cifrado: 26

Texto cifrado: 26
$P = 26^{37} = 5 \bmod 77$
Texto às claras: 5

Exemplo 10.8

Aqui, fornecemos um exemplo mais realista, calculado usando um programa de computador em Java. Escolhemos valores de p e q de 512 *bits** e calculamos n e $\phi(n)$. Em seguida, escolhemos e e calculamos d. Finalmente, mostramos os resultados da cifração e da decifração. O inteiro p é um número de 159 dígitos.

$p =$	96130345313583504574191581280615427909309845594996215822583150879647940455056470638491257160180347503120986666064924201918087806674210 96063354219926661209

* N. de T.: Atualmente, o tamanho de chave RSA de 2.048 *bits* é o mínimo para ser considerado seguro. Recomenda-se o uso de chaves RSA com no mínimo 3.072 *bits* para dados que necessitem ficar armazenados até depois do ano 2030. Mais informações em http://csrc.nist.gov/publications/nistpubs/800-131A/sp800-131A.pdf.

O inteiro q é um número de 160 dígitos.

q =	120601919572314469182767942044508960015559250546370339360617983217314821484837646592153894532091752252732268301071206956046025138871455249690003596600456 17

O módulo é $n = p \times q$. Ele tem 309 dígitos.

n =	1159350417396761496889250986461588752377145737545414477548552613761 4788540832635081727687881596832516846884930062548576411125016241455 2339182927162507656772727460097082714127730434960500556347274566628 0600999240371029914244722922157727985317270338393813346926841373276 2200096667667183183108837342082344437095 3

$\phi(n) = (n - 1) \times (q - 1)$ tem 309 dígitos.

$\phi(n)$ =	1159350417396761496889250986461588752377145737545414477548552613761 4788540832635081727687881596832516846884930062548576411125016241455 2339182927162507656751054233608492916752034482627988117554787657013 9234444057169895817281960982263610754672118646121713591073586406140 0888517026537727726446734106624385766412 8

Bob escolhe $e = 35535$ (o ideal seria 65537). Ele então calcula d.

e =	35535
d =	5800830286003776393609366128967791759466906208965096218042286611138 0593852822358731706286910030021710859044338402170729869087600611530 6202524959884448047568240966247081485817130463240644077704833134010 8509473852956450719367740611973265574242372176176746207763716420760 0337085333288532144708859551366702948 31

Alice deseja enviar a mensagem "THIS IS A TEST"*, que pode ser transformada em um valor numérico usando o esquema de codificação 00-26 (26 representa o caractere de *espaço*).

P =	1907081826081826002619041819

O texto cifrado calculado por Alice é $C = P^e$, mostrado a seguir.

C =	4753091236462268272063655506105451809423717960704917165232392430544 5296061319932856661784341835911415119741125200568297979457173603610 1278218847892741566090480023507190715277185914975188465888632101148 3541033616578984679683867637337657774656250792805211481418440481418 4430812773059004692874248559166462108656

Bob pode recuperar o texto às claras a partir do texto cifrado calculando $P = C^d$, conforme mostrado a seguir.

P =	1907081826081826002619041819

O texto às claras recuperado é "THIS IS A TEST" após a decodificação.

* N. de T.: "This is a test" pode ser traduzido do inglês como "Isto é um teste".

Aplicações

Embora o RSA possa ser usado para cifrar e decifrar mensagens reais, ele torna-se muito lento se a mensagem for longa. O RSA é, portanto, útil para mensagens curtas. Especificamente, veremos que o RSA é usado em assinaturas digitais e outros sistemas criptográficos que geralmente precisam cifrar uma mensagem curta sem ter acesso a uma chave simétrica. O RSA também é usado para autenticação, conforme veremos mais adiante.

10.3 OUTROS ASPECTOS DE SEGURANÇA

Os sistemas criptográficos que estudamos até agora fornecem confidencialidade. Entretanto, em comunicações modernas, precisamos levar em conta outros aspectos da segurança, como integridade, autenticação de entidades e de mensagens, irretratabilidade e gerenciamento de chaves. Discutiremos brevemente essas questões nesta seção.

10.3.1 Integridade das mensagens

Existem ocasiões nas quais nem sequer precisamos de sigilo, porém precisamos garantir a integridade: a mensagem deve permanecer inalterada. Por exemplo, Alice pode escrever um testamento para distribuir seus bens após sua morte; o testamento não precisa ser cifrado. Após sua morte, qualquer pessoa pode examinar o testamento. A integridade dele, entretanto, precisa ser preservada. Alice não quer que o conteúdo do testamento possa ser alterado.

Mensagem e resumo criptográfico da mensagem

Uma forma de preservar a integridade de um documento é por meio de uma *impressão digital*. Se Alice precisa ter a certeza de que o conteúdo de seu documento não será alterado, ela pode colocar a sua impressão digital na parte inferior do documento. Eva não pode modificar o conteúdo desse documento ou criar um documento falso, porque ela não pode forjar a impressão digital de Alice. Para garantir que o documento não foi alterado, a impressão digital de Alice no documento pode ser comparada à impressão digital de Alice em arquivo. Se elas não forem idênticas, o documento não é de Alice. O equivalente eletrônico do par documento e impressão digital é o par *mensagem* e *resumo criptográfico*. Para preservar a integridade de uma mensagem, a mensagem passa por um algoritmo denominado **função de *hash* criptográfico**. A função cria uma versão comprimida da mensagem, conhecida como **resumo criptográfico**, que pode ser usada como uma impressão digital. Para verificar a integridade de uma mensagem ou documento, Bob executa a função de *hash* criptográfico novamente e compara o novo resumo criptográfico com o anterior. Se eles forem idênticos, Bob tem certeza de que a mensagem original não foi alterada. A Figura 10.16 ilustra a ideia.

Os dois pares (documento, impressão digital) e (mensagem, resumo criptográfico da mensagem) são semelhantes, mas têm algumas diferenças. O documento e a impressão digital são ligados fisicamente. A mensagem e o resumo criptográfico da mensagem podem ser desligados (ou enviados separadamente) e, mais importante, o resumo criptográfico da mensagem precisa ser protegido contra modificações.

O resumo criptográfico da mensagem precisa ser protegido contra modificações.

Funções de *hash*

Uma função de *hash* criptográfico recebe uma mensagem de comprimento arbitrário e cria um resumo criptográfico de comprimento fixo. Todas as funções de *hash* criptográfico precisam criar um

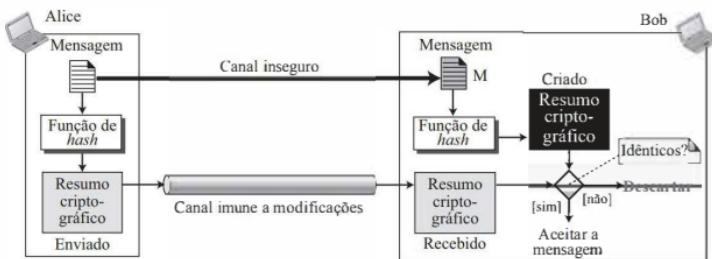



Figura 10.16 Mensagem e resumo criptográfico.

resumo criptográfico de tamanho fixo a partir de uma mensagem de tamanho variável. Para criar uma função com essas características, a melhor forma é por meio de iterações. Em vez de usar uma função de *hash* com uma entrada de tamanho variável, uma função que lida com entradas de tamanho fixo é criada e aplicada repetidamente um número necessário de vezes. A função com entrada de tamanho fixo é conhecida como *função de compressão*. Ela comprime uma cadeia de *n bits* para criar uma cadeia de *m bits*, onde *n* é normalmente maior do que *m*. A construção é conhecida como uma *função de hash criptográfico iterativa*.

Vários algoritmos de *hash* foram desenvolvidos por Ron Rivest. Eles são conhecidos como *MD2*, *MD4* e *MD5*, onde MD significa ***Message Digest*** (Resumo Criptográfico da Mensagem). A última versão, o MD5, é uma versão melhorada do MD4 que divide a mensagem em blocos de 512 *bits* e cria um resumo criptográfico de 128 *bits*. Entretanto, um resumo de mensagem com um tamanho de 128 *bits* é demasiadamente pequeno para resistir a ataques*.

Em resposta à insegurança dos algoritmos de *hash* da família MD, o **Algoritmo de *Hash* Seguro** (SHA – Secure Hash Algorithm) foi inventado. O SHA é um padrão desenvolvido pelo Instituto Nacional de Padrões e Tecnologia (NIST – National Institute of Standards and Technology). O SHA apresenta algumas versões**.

10.3.2 Autenticação de mensagens

Um resumo criptográfico pode ser usado para verificar a integridade de uma mensagem – se ela não foi modificada. Para garantir a integridade da mensagem e autenticar a origem dos dados – que Alice é a autora da mensagem, e não outra pessoa – precisamos incluir no processo um segredo compartilhado por Alice e Bob (desconhecido por Eva); precisamos criar um **Código de Autenticação de Mensagens** (MAC – Message Authentication Code). A Figura 10.17 ilustra a ideia.

Alice usa uma função de *hash* para criar um MAC a partir da concatenação da chave e da mensagem, ***h***(K+M). Ela envia a mensagem e o MAC para Bob através do canal inseguro. Bob separa a mensagem do MAC. Ele gera, então, um novo MAC a partir da concatenação da mensagem e da chave secreta. Bob compara o MAC recém-criado com o recebido. Se os dois MACs são iguais, a mensagem é autêntica e não foi modificada por um adversário.

* N. de T.: As funções de *hash* da família MD apresentam problemas estruturais além dos problemas de tamanho do resumo criptográfico gerado. O seu uso na atualidade é considerado inseguro e deve ser evitado.

** N. de T.: As versões existentes atualmente são SHA-1 (cujo uso em aplicações futuras deve ser evitado), SHA-2 (com tamanhos de *hash* maiores do que o SHA-1) e SHA-3 (o novo padrão, recomendado para aplicações atuais e futuras).

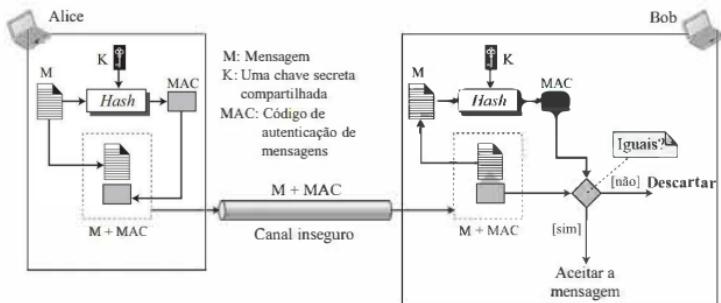



Figura 10.17 Código de Autenticação de Mensagens (MAC).

Perceba que não é necessário utilizar dois canais nesse caso. A mensagem e o MAC podem ser enviados pelo mesmo canal inseguro. Eva pode ver a mensagem, mas não consegue forjar uma nova mensagem para substituí-la porque não possui a chave secreta compartilhada entre Alice e Bob. Ela não é capaz de criar o mesmo MAC que Alice criou.

Um MAC fornece integridade e autenticação de mensagens usando uma combinação de uma função *hash* e de uma chave secreta*.

HMAC

O Instituto Nacional de Padrões e Tecnologia (NIST – National Institute of Standards and Technology) padronizou um algoritmo de MAC conhecido como **HMAC** (*hashed MAC*, ou MAC baseado em *hash*). A implementação do HMAC é muito mais complexa do que a do MAC simplificado descrito anteriormente e não é coberta neste texto.

10.3.3 Assinatura digital

Outra forma de fornecer integridade de mensagens e autenticação de mensagens (e alguns serviços de segurança adicionais, conforme veremos mais adiante) é por meio de uma assinatura digital. Um MAC usa uma chave secreta para proteger o resumo criptográfico; uma assinatura digital usa um par de chaves pública e privada.

Uma assinatura digital usa um par de chaves pública e privada.

Todos temos familiaridade com o conceito de assinatura. Uma pessoa assina um documento para mostrar que ele foi criado ou aprovado por ela. A assinatura é a prova enviada ao destinatário de que o documento vem da entidade correta. Quando um cliente assina um cheque, o banco precisa

* N. de T.: Existem também algoritmos de MAC que não são baseados em funções de hash. Esse é o caso do CMAC, descrito em http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf, que é baseado em uma cifra de bloco.

ter certeza de que o cheque foi emitido por aquele cliente e por mais ninguém. Em outras palavras, uma assinatura em um documento, quando verificada, é um sinal de autenticidade – o documento é autêntico. Considere uma pintura assinada por um artista. A assinatura na arte, se autêntica, indica que a pintura é provavelmente autêntica.

Quando Alice envia uma mensagem a Bob, ele precisa verificar a autenticidade do remetente; precisa ter certeza de que a mensagem veio de Alice, não de Eva. Bob pode pedir a Alice que assine a mensagem eletronicamente. Em outras palavras, uma assinatura eletrônica pode provar a autenticidade de Alice como a remetente da mensagem. Esse tipo de assinatura é conhecido como **assinatura digital**.

Comparação

Observemos as diferenças entre as assinaturas convencionais e as assinaturas digitais.

Inclusão

Uma assinatura convencional é incluída no documento; ela faz parte do documento. Quando preenchemos um cheque, a assinatura fica no cheque; ela não é um documento em separado. Mas quando assinamos um documento digitalmente, enviamos a assinatura como um documento em separado.

Método de verificação

A segunda diferença entre os dois tipos de assinaturas é o método de verificação delas. Para uma assinatura convencional, quando o destinatário recebe um documento, ele compara a assinatura no documento com uma assinatura em arquivo. Se elas forem iguais, o documento é autêntico. O destinatário precisa ter uma cópia dessa assinatura em arquivo para fazer a comparação. No caso de uma assinatura digital, o destinatário recebe a mensagem e a assinatura. Não existe uma cópia da assinatura armazenada em lugar algum. O destinatário precisa aplicar uma técnica de verificação sobre a combinação (mensagem, assinatura) para verificar a sua autenticidade.

Relação

No caso de uma assinatura convencional, existe normalmente uma relação um para muitos entre uma assinatura e diferentes documentos. Uma pessoa usa a mesma assinatura para assinar diversos documentos. No caso de uma assinatura digital, existe uma relação um para um entre uma assinatura e uma mensagem. Cada mensagem tem a sua própria assinatura. A assinatura de uma mensagem não pode ser usada em outra. Se Bob recebe duas mensagens de Alice, uma após a outra, ele não pode usar a assinatura da primeira mensagem para verificar a segunda. Cada mensagem precisa de uma nova assinatura.

Duplicidade

Outra diferença entre os dois tipos de assinaturas é um atributo conhecido como *duplicidade*. Com uma assinatura convencional, uma cópia do documento assinado pode ser diferenciada do documento original em arquivo. Com uma assinatura digital, não existe tal distinção a não ser que haja no documento um fator temporal (como um carimbo de tempo). Por exemplo, considere que Alice envie um documento instruindo Bob a pagar Eva. Se ela interceptar o documento e a assinatura, pode reenviá-lo mais tarde para receber mais dinheiro de Bob.

Processo

A Figura 10.18 mostra o processo de geração e verificação de uma assinatura digital. O emissor usa um *algoritmo de assinatura* para assinar a mensagem. A mensagem e a assinatura são enviadas para o receptor. O receptor recebe a mensagem e a assinatura, e então aplica o *algoritmo de verificação* sobre a combinação. Se o resultado for verdadeiro, a mensagem é aceita; caso contrário, ela é rejeitada.

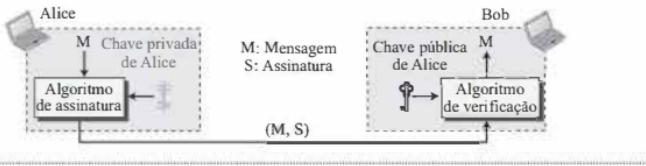



Figura 10.18 Código de Autenticação de Mensagens (MAC).

Uma assinatura convencional é como uma "chave" privada pertencente ao signatário do documento. O signatário a utiliza para assinar documentos; ninguém mais tem essa assinatura. A cópia da assinatura em arquivo é como uma chave pública; qualquer pessoa pode usá-la para verificar um documento, comparando a assinatura desse documento com a assinatura original.

Em uma assinatura digital, o signatário usa sua chave privada, aplicada em um algoritmo de assinatura, para assinar o documento. O verificador, por outro lado, utiliza a chave pública do signatário, aplicada no algoritmo de verificação, para verificar o documento.

Observe que, após um documento ser assinado, qualquer pessoa pode verificá-lo, incluindo Bob, pois todos têm acesso à chave pública de Alice. Ela não deve usar sua chave pública para assinar o documento porque neste caso qualquer pessoa poderia forjar sua assinatura.

Podemos usar uma chave secreta (simétrica) para assinar e verificar uma assinatura? A resposta é não, por diversas razões. Em primeiro lugar, uma chave secreta é conhecida por duas entidades (Alice e Bob, por exemplo). Portanto, se Alice precisar assinar outro documento e enviá-lo para Ted, ela precisa usar outra chave secreta. Em segundo lugar, como veremos mais adiante, a criação de uma chave secreta para uma sessão envolve autenticação, um processo que usa uma assinatura digital. Temos, então, um ciclo vicioso. Em terceiro lugar, Bob poderia usar a chave secreta conhecida por ele e por Alice, assinar um documento, enviá-lo a Ted, e fingir que o documento veio de Alice.

Uma assinatura digital requer um sistema de chave pública.
O signatário assina usando sua chave privada; o verificador verifica usando a chave pública do signatário.

Precisamos fazer uma distinção entre as chaves privada e pública usadas em assinaturas digitais e as chaves pública e privada usadas em um sistema de cifração para fornecer confidencialidade. Neste último caso, as chaves pública e privada do receptor são utilizadas no processo. O emissor usa a chave pública do receptor para cifrar; o receptor usa sua própria chave privada para decifrar. Em uma assinatura digital, as chaves privada e pública do emissor são usadas. O emissor usa sua chave privada; o receptor usa a chave pública do emissor.

Um sistema criptográfico usado para prover confidencialidade utiliza as chaves pública e privada do receptor; uma assinatura digital utiliza as chaves pública e privada do emissor.

Assinando o resumo criptográfico

Afirmamos anteriormente que os sistemas de cifração por chave assimétrica são muito ineficientes quando aplicados a mensagens longas. Em um sistema de assinatura digital, as mensagens normalmente são longas, mas precisamos usar esquemas de chave assimétrica. A solução é assinar um resumo criptográfico da mensagem, que é muito mais curto do que a mensagem em si. Um algoritmo de *hash* cuidadosamente selecionado cria um resumo criptográfico que apresenta uma relação um para um com a mensagem. O remetente pode assinar o resumo criptográfico

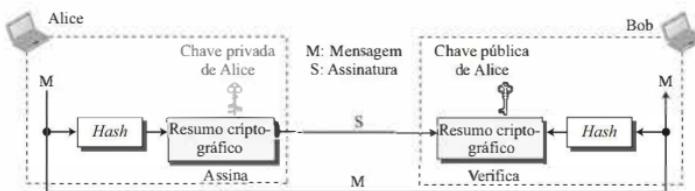



Figura 10.19 Assinando o resumo criptográfico.

da mensagem e o destinatário pode verificar o resumo criptográfico da mensagem. O efeito é o mesmo. A Figura 10.19 mostra o processo de assinar um resumo criptográfico em um sistema de assinatura digital.

Um resumo criptográfico é criado a partir da mensagem no lado de Alice. Em seguida, o resumo criptográfico passa pelo processo de assinatura utilizando a chave privada de Alice, que envia a mensagem e a assinatura para Bob.

No lado de Bob, usando a mesma função de *hash* pública, um resumo criptográfico é primeiramente criado a partir da mensagem recebida. O processo de verificação é aplicado. Se autêntica, a mensagem é aceita; caso contrário, ela é rejeitada.

Serviços

Discutimos diversos serviços de segurança no início do capítulo, incluindo *confidencialidade*, *autenticação* e *integridade de mensagens*, e *irretratabilidade*. Uma assinatura digital pode fornecer diretamente os três últimos serviços; já para a confidencialidade de mensagens, ainda precisamos de esquemas de cifração/decifração.

Autenticação de mensagens

Um esquema de assinatura digital seguro, assim como uma assinatura convencional segura (que não pode ser facilmente copiada) pode fornecer autenticação de mensagens (também conhecida como autenticação da origem dos dados). Bob pode verificar que a mensagem foi enviada por Alice porque a chave pública dela é usada na verificação. A chave pública de Alice não pode verificar a assinatura de um documento assinado pela chave privada de Eva.

Integridade de mensagens

A integridade da mensagem é preservada se assinarmos a mensagem ou o resumo criptográfico da mensagem, pois não podemos obter o mesmo resumo criptográfico se alguma parte da mensagem for alterada. Os esquemas de assinatura digital atuais utilizam uma função de *hash* como parte dos algoritmos de assinatura e de verificação, de modo a preservar a integridade da mensagem.

Irretratabilidade

Se Alice assinar uma mensagem e depois negar tê-lo feito, Bob é capaz de provar posteriormente que ela realmente assinou a mensagem? Por exemplo, se Alice enviar uma mensagem a um banco (Bob) e pedir a transferência de R$ 10.000 de sua conta para a conta de Ted, ela pode negar mais tarde ter enviado essa mensagem? Com o esquema que apresentamos até agora, Bob pode ter um problema. Ele precisa manter a assinatura em arquivo e depois usar a chave pública de Alice para criar a mensagem original com o objetivo de provar que a mensagem em arquivo e a mensagem recém-criada são iguais. Isso não é viável, porque Alice pode ter trocado sua chave privada ou

pública nesse intervalo de tempo; ela pode também afirmar que o arquivo que contém a assinatura não é autêntico.

Uma solução para esse problema é envolver uma terceira parte confiável. Um grupo de pessoas pode definir uma terceira parte que seja confiável por elas. Mais adiante, veremos que uma terceira parte confiável pode resolver muitos outros problemas relativos a serviços de segurança e à troca de chaves. A Figura 10.20 mostra como uma parte confiável pode impedir que Alice negue ter enviado a mensagem.

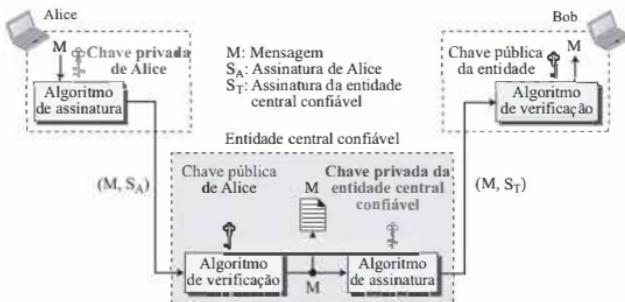



Figura 10.20 Usando uma entidade central confiável para obter irretratabilidade.

Alice cria uma assinatura para sua mensagem (S_A) e envia a mensagem, sua identidade, a identidade de Bob e a assinatura para uma entidade central confiável. A entidade central, após verificar que a chave pública de Alice é válida, verifica por meio dessa chave que a mensagem veio mesmo de Alice. A entidade central, em seguida, salva em seu arquivo uma cópia da mensagem com a identidade da remetente, a identidade do destinatário e um carimbo de tempo. A entidade central utiliza sua chave privada para criar outra assinatura (S_T) para a mensagem. A entidade central, em seguida, envia a mensagem, a nova assinatura, a identidade de Alice e a identidade de Bob para Bob, que verifica a mensagem usando a chave pública da entidade central confiável.

Se no futuro Alice negar que tenha enviado a mensagem, a entidade central pode mostrar uma cópia da mensagem armazenada. Se a mensagem de Bob for uma duplicata da mensagem armazenada pela entidade central, Alice perderá a disputa. Para que todo o processo seja confidencial, um nível de cifração/decifração pode ser adicionado ao sistema, conforme discutido na próxima seção.

Confidencialidade

Uma assinatura digital não fornece confidencialidade da comunicação. Se confidencialidade for necessária, a mensagem e a assinatura devem ser cifradas usando uma cifra de chave simétrica ou assimétrica.

Esquema de assinatura digital RSA

Diversos *esquemas de assinatura digital* evoluíram durante as últimas décadas. Alguns deles foram implantados. Nesta seção, descrevemos brevemente um deles, o RSA. Em uma seção anterior, discutimos como usar o sistema criptográfico RSA para prover confidencialidade. A ideia do RSA pode também ser utilizada para assinar e verificar uma mensagem. Nesse caso, ele é denominado *esquema de assinatura digital RSA*. O esquema de assinatura digital troca os papéis das chaves pública e privada. Em primeiro lugar, as chaves pública e privada do remetente são usadas, não

as do destinatário. Em segundo lugar, o remetente usa sua própria chave privada para assinar o documento; o destinatário usa a chave pública do remetente para verificar a assinatura. Se compararmos esse esquema com a maneira convencional de assinar, veremos que a chave privada desempenha o papel da assinatura do remetente, e a chave pública do remetente desempenha o papel da cópia da assinatura que fica disponível ao público. Obviamente, Alice não pode usar a chave pública de Bob para assinar a mensagem, porque, nesse caso, qualquer outra pessoa poderia fazer o mesmo. Os locais onde é feita a assinatura e onde é feita a verificação usam a mesma função, mas com parâmetros diferentes. O verificador compara a mensagem e a saída da função, verificando se os dois valores são iguais usando aritmética modular. Se for o caso, a mensagem é aceita. A Figura 10.21 mostra o esquema no qual a assinatura e a verificação são aplicadas sobre o resumo criptográfico da mensagem em vez da mensagem em si, pois a criptografia de chave pública não é muito eficiente para uso com mensagens longas; o resumo criptográfico é muito menor do que a própria mensagem.

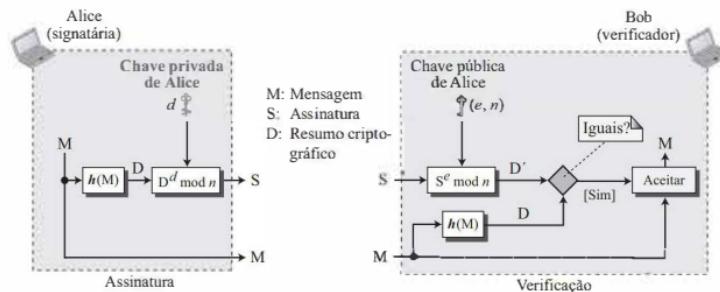



Figura 10.21 A assinatura RSA sobre o resumo criptográfico da mensagem.

Alice, a signatária, utiliza primeiramente uma função de *hash* previamente acordada para criar um resumo criptográfico da mensagem, D = ***h***(M). Ela, então, assina o resumo criptográfico, gerando a assinatura $S = D^d \bmod n$. A mensagem e a assinatura são enviadas para Bob. Ele o verificador, recebe a mensagem e a assinatura e usa primeiramente o expoente público de Alice para recuperar o resumo criptográfico, $D' = S^e \bmod n$. Ele, então, aplica o algoritmo de *hash* sobre a mensagem recebida para obter D = ***h***(M). Bob agora compara os dois resumos criptográficos, D e D'. Se eles forem iguais (usando aritmética modular), ele aceita a mensagem.

Digital Signature Standard

O **Padrão de Assinatura Digital** (DSS – Digital Signature Standard), foi aprovado pelo NIST em 1994. O DSS é um esquema de assinatura digital complexo e mais seguro.

10.3.4 Autenticação de entidades

A autenticação de entidades é uma técnica projetada para permitir que uma parte verifique a identidade de outra parte. Uma *entidade* pode ser uma pessoa, um processo, um cliente ou um servidor. A entidade cuja identidade precisa ser provada é chamada de *requerente*; a parte que tenta verificar a identidade do requerente é conhecida como *verificador*.

Autenticação de entidades *versus* autenticação de mensagens

Existem duas diferenças entre a *autenticação de entidades* e a *autenticação de mensagens* (*autenticação da origem dos dados*).

1. A autenticação de mensagens (ou autenticação da origem dos dados) não precisa necessariamente acontecer em tempo real; já a autenticação de entidades precisa. No primeiro caso, Alice envia uma mensagem para Bob. Quando Bob autentica a mensagem, Alice pode ou não estar presente no processo de comunicação. Por outro lado, quando a autenticação de entidades por parte de Alice é necessária, nenhuma mensagem é de fato transmitida até que Alice seja autenticada por Bob. Ela precisa estar conectada e participar do processo. Apenas após Alice ser autenticada as mensagens podem ser trocadas entre ela e Bob. A autenticação da origem dos dados é necessária quando um e-mail é enviado de Alice para Bob. A autenticação de entidades é necessária quando Alice recebe dinheiro em um caixa eletrônico.
2. A autenticação de mensagens simplesmente autentica uma mensagem; o processo precisa ser repetido para cada nova mensagem. A autenticação de entidades autentica o requerente por toda a duração de uma sessão.

Categorias de verificação

Na autenticação de entidades, o requerente deve identificar-se junto ao verificador. Isto pode ser feito com um dentre três tipos de prova de identidade: *algo conhecido*, *algo possuído* ou *algo inerente*.

- **Algo conhecido.** Refere-se a um segredo conhecido apenas pelo requerente que pode ser verificado pelo verificador. Exemplos incluem uma senha, um PIN, uma chave secreta ou uma chave privada.
- **Algo possuído.** Refere-se a algo que pode provar a identidade do requerente. Exemplos incluem um passaporte, uma carteira de motorista, um cartão de identificação, um cartão de crédito e um *smart card*.
- **Algo inerente.** Refere-se a uma característica inerente ao requerente. Exemplos incluem assinaturas convencionais, impressões digitais, voz, características faciais, padrões de retina e caligrafia.

Nesta seção, discutimos apenas o primeiro tipo de prova de identidade, *algo conhecido*, que é normalmente utilizado para autenticação remota (via rede) de entidades. As outras duas categorias são normalmente utilizadas quando o requerente está presente em pessoa.

Senhas

O método mais simples e mais antigo de autenticação de entidades é o uso de uma senha, que é algo que o requerente *conhece*. A senha é usada quando um usuário precisa acessar os recursos de um sistema (*login*). Cada usuário tem uma identificação de usuário pública e uma senha privada. Senhas, no entanto, são muito propensas a ataques. A senha pode ser roubada, interceptada, adivinhada, e assim por diante.

Desafio-resposta

Na autenticação por senha, o requerente prova sua identidade demonstrando que ele sabe um segredo, a senha. No entanto, caso o requerente envie esse segredo, ele ficará suscetível à interceptação por atacantes. Na **autenticação por desafio-resposta**, o requerente prova que ele *sabe* um segredo

sem enviá-lo. Em outras palavras, o requerente não envia o segredo para o verificador; o verificador possui esse segredo ou é capaz de determiná-lo.

Em uma autenticação por desafio-resposta, o requerente prova que ele sabe um segredo sem enviá-lo ao verificador.

O *desafio* é um valor que varia com o tempo, como um número aleatório ou um carimbo de tempo, enviado pelo verificador. O requerente aplica uma função sobre o desafio e envia o resultado, chamado de *resposta*, ao verificador. A resposta mostra que o requerente conhece o segredo.

Usando uma cifra de chave simétrica

Diversas abordagens de autenticação por desafio-resposta usam criptografia de chave simétrica. O segredo nesse caso é a chave secreta compartilhada, conhecida tanto pelo requerente como pelo verificador. A função é o algoritmo de cifração aplicado sobre o desafio. Embora existam várias abordagens para implementar esse método, mostramos apenas a mais simples para dar uma ideia de seu funcionamento. A Figura 10.22 mostra essa primeira abordagem.

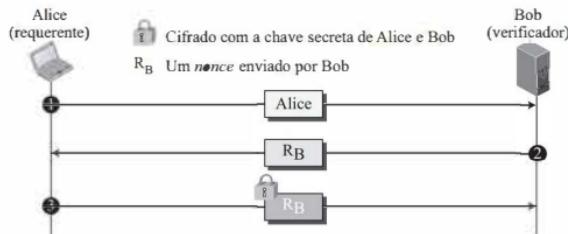



Figura 10.22 Autenticação unidirecional por chave simétrica.

A primeira mensagem não faz parte do processo de desafio-resposta; ela apenas informa o verificador que o requerente quer ser desafiado. A segunda mensagem é o desafio. R_B é o *nonce* (que significa número usado apenas uma vez) escolhido aleatoriamente pelo verificador (Bob) para desafiar o requerente. O requerente cifra o *nonce* usando a chave secreta compartilhada conhecida apenas pelo requerente e pelo verificador e envia o resultado para o verificador. O verificador decifra a mensagem. Se o *nonce* obtido a partir de decifração for idêntico àquele enviado pelo verificador, Alice tem seu acesso permitido.

Perceba que, nesse processo, o requerente e o verificador precisam manter a chave simétrica utilizada no processo em sigilo. O verificador deve também armazenar o valor do *nonce* para a identificação do requerente até que a resposta seja retornada.

Usando uma cifra de chave assimétrica

A Figura 10.23 mostra essa abordagem.

Em vez de uma cifra de chave simétrica, podemos usar uma cifra de chave assimétrica para a autenticação de entidades. Nesse caso, o segredo deve ser a chave privada do requerente. O requerente deve demonstrar que ele possui a chave privada relacionada à chave pública que fica disponível para todos. Isso significa que o verificador deve cifrar o desafio usando a chave pública do requerente; o requerente, então, decifra a mensagem utilizando sua chave privada. A resposta ao desafio é a mensagem decifrada.

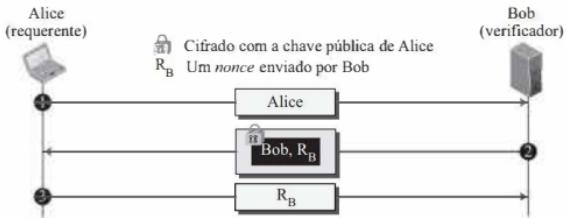



Figura 10.23 Autenticação unidirecional por chave assimétrica.

Usando assinaturas digitais

A autenticação de entidades pode também ser feita usando uma assinatura digital. Quando uma assinatura digital é utilizada para autenticar entidades, o requerente utiliza sua chave privada para gerar uma assinatura. Em uma primeira abordagem, mostrada na Figura 10.24, Bob usa um texto às claras como desafio e Alice assina a resposta.

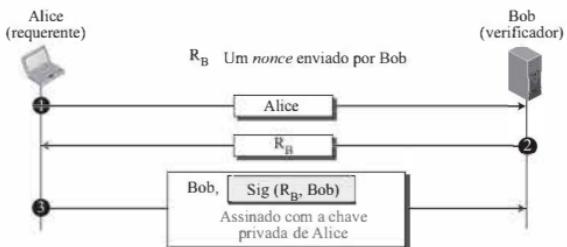



Figura 10.24 Autenticação unidirecional por assinatura digital.

10.3.5 Gerenciamento de chaves

Discutimos criptografia de chave simétrica e de chave assimétrica nas seções anteriores. No entanto, ainda não discutimos como as chaves secretas usadas na criptografia de chave simétrica são distribuídas e gerenciadas, nem como isso ocorre para as chaves públicas na criptografia de chave assimétrica. Esta seção aborda essas duas questões.

Distribuição de chaves simétricas

A criptografia de chave simétrica é mais eficiente do que a criptografia de chave assimétrica para cifrar mensagens grandes. A criptografia de chave simétrica, no entanto, requer uma chave secreta compartilhada entre as duas partes.

Se Alice precisa trocar mensagens confidenciais com N pessoas, ela precisa de N chaves diferentes. O que acontece se N pessoas precisarem se comunicar umas com as outras? Um total de $N(N - 1)$ chaves são necessárias se precisarmos que duas pessoas usem duas chaves para estabelecer

uma comunicação bidirecional; apenas $N(N-1)/2$ chaves são necessárias se permitirmos que uma mesma chave seja usada em ambas as direções. Isto significa que, se um milhão de pessoas precisarem se comunicar umas com as outras, cada pessoa precisará ter um milhão de chaves diferentes; no total, meio trilhão de chaves serão necessárias. Isto é normalmente conhecido como o problema N^2 porque o número de chaves necessárias para N entidades é próximo de N^2.

O número de chaves não é o único problema; a distribuição de chaves é outro problema. Se Alice e Bob querem se comunicar, eles precisam de algum método para compartilhar uma chave secreta; se Alice deseja se comunicar com um milhão de pessoas, como ela pode trocar um milhão de chaves com um milhão de pessoas? Usar a Internet definitivamente não é um método seguro. É óbvio que precisamos de uma forma eficiente de gerenciar e distribuir chaves secretas.

Centro de Distribuição de Chaves: KDC

Uma solução prática é envolver uma terceira parte confiável, conhecida como **Centro de Distribuição de Chaves** (KDC – Key Distribution Center). Para reduzir o número de chaves, cada pessoa estabelece uma chave secreta compartilhada com o KDC. Uma chave secreta é estabelecida entre o KDC e cada membro. Agora, a questão é como Alice pode enviar uma mensagem confidencial para Bob. O processo é o seguinte:

1. Alice envia uma solicitação para o KDC, afirmando que ela precisa de uma chave secreta de sessão (temporária) entre ela e Bob.
2. O KDC informa Bob sobre a solicitação de Alice.
3. Se Bob concordar, uma chave de sessão é criada entre os dois.

As chaves secretas compartilhadas entre Alice e o KDC e entre Bob e o KDC são usadas para autenticar Alice e Bob em relação ao KDC e para impedir que Eva personifique qualquer um deles.

Múltiplos KDCs Quando o número de pessoas que utilizam um KDC aumenta, o gerenciamento do sistema pode ficar difícil e um gargalo pode aparecer. Para resolver o problema, precisamos ter múltiplos KDCs. Podemos dividir o mundo em domínios. Cada domínio pode ter um ou mais KDCs (para redundância em caso de falha). Nesse cenário, se Alice quiser enviar uma mensagem confidencial para Bob, que pertence a outro domínio, Alice entra em contato com o KDC dela, que entra, então, em contato com o KDC no domínio de Bob. Os dois KDCs podem criar uma chave secreta entre Alice e Bob. Pode haver KDCs locais, KDCs nacionais e KDCs internacionais. Quando Alice precisar se comunicar com Bob, que vive em outro país, ela envia seu pedido a um KDC local; o KDC local encaminha o pedido para o KDC nacional; o KDC nacional encaminha o pedido para um KDC internacional. O pedido é, então, encaminhado até o KDC local onde Bob vive. A Figura 10.25 mostra uma configuração de múltiplos KDCs hierárquicos.

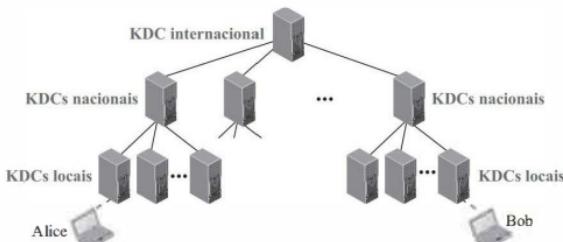



Figura 10.25 Múltiplos KDCs.

Chaves de sessão Um KDC cria uma chave secreta para cada membro; ela pode ser usada apenas entre o membro e o KDC, não entre dois membros. Se Alice precisa se comunicar secretamente com Bob, ela precisa de uma chave secreta entre ela e Bob. Um KDC pode criar uma *chave de sessão* entre Alice e Bob, usando suas chaves compartilhadas com o KDC. As chaves de Alice e Bob são usadas para autenticar Alice e Bob em relação ao KDC e em relação um ao outro antes que a chave de sessão seja estabelecida. Após o término da comunicação, a chave de sessão deixa de ser útil.

Uma chave de sessão simétrica entre duas partes é usada durante uma única sessão.

Diversas abordagens diferentes foram propostas para criar a chave da sessão usando ideias discutidas nas seções anteriores. Mostramos a abordagem mais simples na Figura 10.26. Embora essa abordagem seja muito rudimentar, ela ajuda a compreender abordagens mais sofisticadas presentes na literatura.

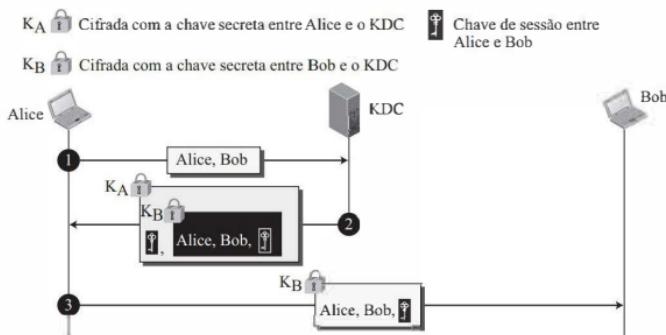



Figura 10.26 Criando uma chave de sessão usando um KDC.

1. Alice envia uma mensagem às claras ao KDC para obter uma chave de sessão simétrica entre ela e Bob. A mensagem contém a sua identidade registrada (a palavra *Alice* na figura) e a identidade de Bob (a palavra *Bob* na figura). Essa mensagem não é cifrada, mas sim pública. O KDC não se importa.
2. O KDC recebe a mensagem e cria algo conhecido como **bilhete** ou ***ticket***. O bilhete é cifrado usando a chave de Bob (K_B) e contém as identidades de Alice e de Bob, bem como a chave de sessão. O bilhete com uma cópia da chave de sessão é enviado para Alice. Ela recebe a mensagem, a decifra, e extrai a chave de sessão. Ela não pode decifrar o bilhete de Bob; o bilhete foi criado para Bob, não para Alice. Perceba que essa mensagem contém uma cifração dupla – o bilhete é cifrado, e toda a mensagem também é cifrada. Na segunda mensagem, Alice é de fato autenticada com relação ao KDC, pois apenas Alice pode abrir a mensagem completa usando sua chave secreta compartilhada com o KDC.
3. Alice envia o bilhete para Bob, que abre o bilhete e sabe que Alice precisa enviar mensagens para ele usando a chave de sessão. Perceba que, nessa mensagem, Bob é autenticado com relação ao KDC porque apenas Bob pode abrir o bilhete. Como Bob é autenticado em relação ao KDC, ele também é autenticado em relação a Alice, que confia no KDC. Da mesma forma, Alice também é autenticada em relação a Bob, pois Bob confia no KDC e o KDC enviou a Bob o bilhete que inclui a identidade de Alice.

Acordo de chave simétrica

Alice e Bob podem criar uma chave de sessão entre eles sem usar um KDC. Esse método de geração de chaves de sessão é conhecido como *acordo de chave simétrica*. Apesar de existirem diversas maneiras de se conseguir fazer isso, discutimos apenas um método, denominado Diffie-Hellman, que mostra a ideia básica utilizada em métodos mais sofisticados (menos propensos a ataques).

Acordo de chaves Diffie-Hellman

No **protocolo Diffie-Hellman**, duas partes criam uma chave de sessão simétrica sem precisar de um KDC. Antes de estabelecer uma chave simétrica, as duas partes precisam escolher dois números p e g. Esses dois números apresentam algumas propriedades discutidas na teoria dos números, mas essa discussão não faz parte do escopo deste livro. Os dois números não precisam ser confidenciais. Eles podem ser enviados via Internet; podem ser públicos. A Figura 10.27 mostra o procedimento.

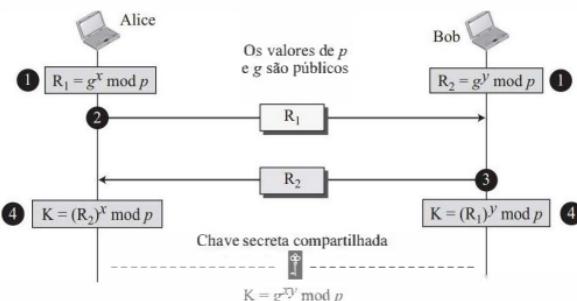



Figura 10.27 Método de Diffie-Hellman.

Os passos envolvidos são os seguintes:

1. Alice escolhe um número aleatório grande x tal que $0 \leq x \leq p - 1$ e calcula $R_1 = g^x \bmod p$. Bob escolhe outro número aleatório grande y tal que $0 \leq y \leq p - 1$ e calcula $R_2 = g^y \bmod p$.
2. Alice envia R_1 para Bob. Perceba que Alice não envia o valor de x; ela envia apenas R_1.
3. Bob envia R_2 para Alice. Novamente, perceba que Bob não envia o valor de y, mas apenas R_2.
4. Alice calcula $K = (R_2)^x \bmod p$. Bob também calcula $K = (R_1)^y \bmod p$.

K é a chave simétrica para a sessão.

$$K = (g^x \bmod p)^y \bmod p = (g^y \bmod p)^x \bmod p = g^{xy} \bmod p$$

Bob calculou $K = (R_1)^y \bmod p = (g^x \bmod p)^y \bmod p = g^{xy} \bmod p$. Alice calculou $K = (R_2)^x \bmod p = (g^y \bmod p)^x \bmod = g^{xy} \bmod p$. Ambos chegaram ao mesmo valor apesar de Bob não saber o valor de x e de Alice não saber o valor de y.

A chave simétrica (compartilhada) no protocolo Diffie-Hellman é $K = g^{xy} \bmod p$.

Exemplo 10.9

Mostraremos um exemplo trivial para tornar esse procedimento mais claro. Nosso exemplo usa números pequenos, porém, note que, em uma situação real, os números são muito grandes. Considere que $g = 7$ e $p = 23$. Os passos envolvidos são os seguintes:

1. Alice escolhe $x = 3$ e calcula $R_1 = 7^3 \bmod 23 = 21$. Bob escolhe $y = 6$ e calcula $R_2 = 7^6 \bmod 23 = 4$.
2. Alice envia o número 21 para Bob.
3. Bob envia o número 4 para Alice.
4. Alice calcula a chave simétrica $K = 4^3 \bmod 23 = 18$. Bob calcula a chave simétrica $K = 21^6 \bmod 23 = 18$.
   O valor de K é o mesmo tanto para Alice como para Bob; $g^{xy} \bmod p = 7^{18} \bmod 23 = 18$.

Distribuição de chaves públicas

Na criptografia de chave assimétrica, as pessoas não precisam conhecer uma chave simétrica compartilhada. Se Alice quiser enviar uma mensagem para Bob, ela só precisa saber a chave pública de Bob, a qual é aberta ao público e disponível para todos. Se Bob precisar enviar uma mensagem para Alice, ele só precisa saber a chave pública de Alice, que também é conhecida por todos. Na criptografia de chave pública, todos protegem uma chave privada e divulgam uma chave pública.

Na criptografia de chave pública, todos têm acesso à chave pública de todos; as chaves públicas ficam disponíveis ao público.

As chaves públicas, assim como as chaves secretas, precisam ser distribuídas para serem úteis. Discutiremos brevemente como as chaves públicas podem ser distribuídas.

Anúncio público

A abordagem simplista consiste em anunciar publicamente as chaves públicas. Bob pode colocar a sua chave pública em seu *site* ou anunciá-la em um jornal local ou nacional. Quando Alice precisar enviar uma mensagem confidencial para Bob, ela pode obter a chave pública de Bob a partir de seu *site* ou do jornal, ou mesmo enviando uma mensagem requisitando tal chave. Essa abordagem, no entanto, não é segura, pois é sujeita à falsificação. Por exemplo, Eva poderia fazer tal anúncio público. Antes que Bob pudesse reagir, o dano já poderia ter sido causado. Eva poderia convencer Alice a lhe enviar uma mensagem destinada a Bob. Eva também poderia assinar um documento com uma chave privada forjada correspondente e fazer com que todos acreditassem que ela foi assinada por Bob. A abordagem também é vulnerável se Alice pede a chave pública de Bob diretamente a ele. Eva pode interceptar a resposta de Bob e substituir a chave pública de Bob pela sua própria chave pública forjada.

Autoridade certificadora

A abordagem geralmente empregada para a distribuição de chaves públicas é a criação de **certificados de chave pública**, também conhecidos como **certificados digitais**. Bob deseja duas coisas: quer que as pessoas conheçam a sua chave pública e também que ninguém aceite uma chave pública forjada como sendo sua. Bob pode ir a uma **Autoridade Certificadora** (AC), uma organização federal ou estadual que vincula uma chave pública a uma entidade e emite um certificado. A Figura 10.28 ilustra esse conceito.

A AC em si tem uma chave pública bem conhecida que não pode ser forjada. A AC verifica a identificação de Bob (usando uma identidade com foto, juntamente com outros comprovantes). Ela,

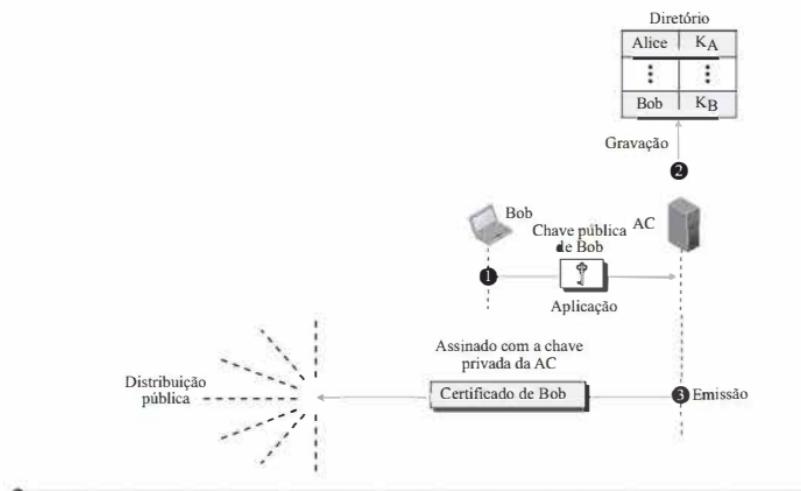



Figura 10.28 Autoridade Certificadora.

então, pede a chave pública de Bob e a insere em um certificado. Para evitar que o próprio certificado seja forjado, a AC assina o certificado com sua chave privada. Agora, Bob pode distribuir o certificado assinado. Quem quiser saber a chave pública de Bob obtém o certificado assinado e usa a chave pública da autoridade para extrair a chave pública de Bob.

X.509

Embora o uso de uma AC tenha resolvido o problema de fraudes da chave pública, essa estratégia criou um efeito colateral. Cada certificado pode ter um formato diferente. Se Alice quiser usar um programa para obter automaticamente diferentes certificados e resumos criptográficos pertencentes a pessoas distintas, o programa pode não ser capaz de fazer isso. Um certificado pode apresentar a chave pública em um formato e outro certificado pode apresentá-la em um formato diferente. A chave pública pode estar na primeira linha de um certificado e na terceira linha de outro. Qualquer coisa que precisa ser utilizada universalmente deve apresentar um formato universal. Para remover esse efeito colateral, a ITU desenvolveu o **X.509**, uma recomendação que foi incorporada pela Internet com algumas alterações. O X.509 é um modo de descrever o certificado de forma estruturada. Ele usa um protocolo bem conhecido chamado ASN.1 (discutido no Capítulo 9) que define campos familiares a programadores de computador.

10.4 SEGURANÇA DA INTERNET

Nesta seção, discutimos como os princípios da criptografia são aplicados à Internet, e também segurança na camada de aplicação, de transporte e de rede. A segurança na camada de enlace de dados é normalmente uma questão proprietária e é implementada pelos projetistas de LANs e WANs.

10.4.1 Segurança na camada de aplicação

Esta seção discute dois protocolos que fornecem serviços de segurança para mensagens de correio eletrônico (e-mail): Privacidade Bastante Boa (PGP – Pretty Good Privacy) e Extensões Seguras/Multifunção para Mensagens de Internet (S/MIME – Secure/Multipurpose Internet Mail Extension).

Segurança de e-mail

O envio de um e-mail é uma atividade não interativa. A natureza dessa atividade é diferente das que veremos nas próximas duas seções: o SSL e o IPSec. Nesses protocolos, consideramos que as duas partes criam uma sessão entre elas e trocam dados em ambas as direções. No e-mail, não existe uma sessão. Alice e Bob não podem criar uma sessão. Alice envia uma mensagem para Bob; algum tempo depois, Bob lê a mensagem e pode ou não enviar uma resposta. Discutimos a segurança de uma mensagem unidirecional porque aquilo que Alice envia para Bob é totalmente independente daquilo que Bob envia para Alice.

Algoritmos criptográficos

Se o e-mail é uma atividade não interativa, como o remetente e o destinatário podem chegar a um acordo sobre o algoritmo criptográfico a ser usado para fornecer segurança ao e-mail? Se não há uma sessão nem um protocolo para negociar os algoritmos de cifração/decifração e de *hash*, como o destinatário pode saber qual algoritmo o remetente escolheu para cada propósito?

Para resolver esse problema, o protocolo define um conjunto de algoritmos para cada operação que o usuário utilizou em seu sistema. Alice inclui os nomes (ou identificadores) dos algoritmos que ela usou no próprio e-mail. Por exemplo, Alice pode escolher o AES para a cifração/decifração e o SHA-2 para calcular o *hash*. Quando Alice enviar uma mensagem para Bob, ela inclui os identificadores correspondentes ao AES e ao SHA-2 em sua mensagem. Bob recebe a mensagem e, primeiramente, extrai os identificadores. Ele, então, sabe qual algoritmo usar para decifrar a mensagem e qual usar para calcular o seu resumo criptográfico.

Na segurança de e-mails, o remetente da mensagem deve incluir na mensagem os nomes ou identificadores dos algoritmos utilizados.

Segredos criptográficos

O mesmo problema encontrado com os algoritmos de criptografia se aplica aos segredos criptográficos (chaves). Se não há uma fase de negociação, como as duas partes estabelecem segredos entre elas? Os protocolos de segurança de e-mail atuais exigem que a cifração e a decifração sejam feitas usando um algoritmo de chave simétrica e uma chave secreta utilizada uma única vez, enviada juntamente com a mensagem. Alice pode criar uma chave secreta e enviá-la com a mensagem para Bob. Para proteger a chave secreta contra uma possível interceptação por Eva, a chave secreta é cifrada com a chave pública de Bob. Em outras palavras, a própria chave secreta é cifrada.

Na de segurança e-mails, a cifração e a decifração são feitas usando um algoritmo de chave simétrica, mas a chave secreta para decifrar a mensagem é cifrada usando a chave pública do destinatário e é enviada juntamente com a mensagem.

Certificados

Outra questão precisa ser considerada antes de discutirmos qualquer protocolo de segurança de e-mail em particular. É óbvio que alguns algoritmos de chave pública devem ser usados para fornecer segurança ao e-mail. Por exemplo, precisamos cifrar a chave secreta ou assinar a mensagem. Para cifrar a chave secreta, Alice precisa da chave pública de Bob; para verificar uma mensagem assinada, Bob precisa da chave pública de Alice. Portanto, para enviar uma pequena mensagem autenticada e confidencial, duas chaves públicas são necessárias. Como Alice pode ter certeza de qual é a chave pública de Bob, e como Bob pode ter a certeza de qual é a chave pública de Alice? Cada protocolo de segurança de e-mail adota um método diferente para certificar chaves.

Pretty Good Privacy

O primeiro protocolo discutido nesta seção é conhecido como **Privacidade Bastante Boa** (PGP – Pretty Good Privacy). O PGP foi inventado por Phil Zimmermann para prover confidencialidade, integridade e autenticidade aos e-mails. O PGP pode ser usado para criar mensagens de e-mail seguras.

Cenários

Primeiramente, discutiremos a ideia geral do PGP, partindo de um cenário simples e chegando a outro complexo. Usamos o termo "Dados" para nos referirmos à mensagem antes do processamento.

Texto às claras O cenário mais simples é enviar a mensagem de e-mail na forma de texto às claras, conforme mostra a Figura 10.29. Não há integridade ou confidencialidade da mensagem nesse cenário.

Figura 10.29 Uma mensagem às claras.

Integridade das mensagens Provavelmente, o aperfeiçoamento seguinte permite que Alice assine a mensagem. Ela cria um resumo criptográfico da mensagem e assina o resumo usando sua chave privada. A Figura 10.30 ilustra essa situação.

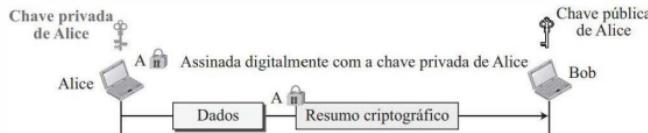



Figura 10.30 Uma mensagem autenticada.

Quando Bob recebe a mensagem, ele a verifica usando a chave pública de Alice. Duas chaves são necessárias para esse cenário. Alice precisa saber sua chave privada; já Bob precisa saber a chave pública de Alice.

Compressão Uma melhoria adicional é comprimir para tornar o pacote mais compacto. Essa melhoria não resulta em um benefício de segurança, mas alivia o tráfego. A Figura 10.31 mostra esse novo cenário.

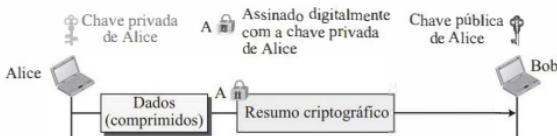



Figura 10.31 Uma mensagem comprimida.

Confidencialidade com uma chave de sessão usada uma única vez A Figura 10.32 mostra essa situação. Conforme discutimos anteriormente, o sigilo em um sistema de e-mail pode ser obtido aplicando-se cifração convencional com uma chave de sessão sendo usada uma única vez. Alice pode criar uma chave de sessão, usá-la para cifrar a mensagem e calcular o resumo criptográfico, e enviar a própria chave juntamente com a mensagem. Entretanto, para proteger a chave de sessão, Alice cifra essa chave com a chave pública de Bob.

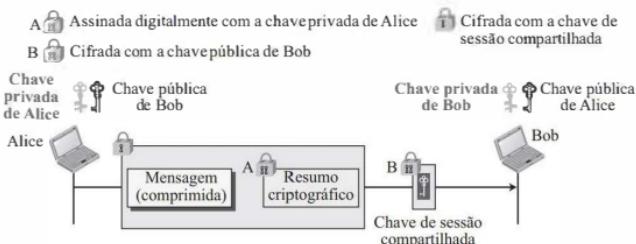



Figura 10.32 Uma mensagem confidencial.

Quando Bob recebe o pacote, primeiramente decifra a chave de sessão, usando sua chave privada. Então utiliza a chave de sessão para decifrar o restante da mensagem. Após descomprimir o restante da mensagem, Bob calcula o resumo criptográfico da mensagem e verifica se ele é igual ao resumo criptográfico enviado por Alice. Se for o caso, então a mensagem é autêntica.

Conversão de código Outro serviço prestado pelo PGP é a conversão de código. A maioria dos sistemas de e-mail permite que a mensagem seja composta apenas por caracteres ASCII. Para traduzir outros caracteres que não pertençam ao conjunto ASCII, o PGP usa o esquema de conversão Base64 (ver Capítulo 2).

Segmentação

O PGP permite a segmentação da mensagem após ela ter sido convertida usando Base64 para possibilitar que cada unidade transmitida tenha o tamanho uniforme permitido pelo protocolo de e-mail subjacente.

Molhos de chaves

Em todos os cenários anteriores, consideramos que Alice precisava enviar uma mensagem apenas para Bob. Esse nem sempre é o caso. Alice pode precisar enviar mensagens para muitas pessoas; ela precisa de *molhos de chaves*. Nesse caso, Alice precisa de um molho de chaves públicas, com uma chave pertencente a cada pessoa com quem Alice precisa se corresponder (enviar ou receber mensagens). Além disso, os projetistas do PGP especificaram um molho de chaves públicas e privadas. Uma razão é que Alice pode desejar alterar o seu par de chaves de tempos em tempos. Outra razão é que Alice pode precisar se corresponder com diferentes grupos de pessoas (amigos, colegas e assim por diante). Alice pode querer usar um par de chaves diferente para cada grupo. Portanto, cada usuário precisa ter dois conjuntos de molhos de chaves: um molho de suas chaves privadas e um molho de chaves públicas de outras pessoas. A Figura 10.33 mostra uma comunidade com três pessoas, cada uma possuindo um molho de pares de chaves públicas e privadas e, ao mesmo tempo, um molho de chaves públicas pertencentes a outras pessoas da comunidade.

Figura 10.33 Molhos de chaves no PGP

Alice, por exemplo, tem vários pares de chaves públicas e privadas pertencentes a ela e chaves públicas pertencentes a outras pessoas. Perceba que todos podem ter mais do que uma chave pública. Dois casos podem surgir.

1. Alice precisa enviar uma mensagem para outra pessoa da comunidade.
   a. Ela usa sua chave privada para assinar o resumo criptográfico da mensagem.
   b. Ela usa a chave pública do destinatário para cifrar uma chave de sessão recém-criada.
   c. Ela usa a chave de sessão criada para cifrar a mensagem e o resumo criptográfico assinado.
2. Alice recebe uma mensagem de outra pessoa da comunidade.
   a. Ela usa sua chave privada para decifrar a chave de sessão.
   b. Ela usa a chave de sessão para decifrar a mensagem e o resumo criptográfico.
   c. Ela usa sua chave pública para verificar o resumo criptográfico.

Algoritmos do PGP

O PGP define um conjunto de algoritmos de chave assimétrica e de chave simétrica, funções de *hash* criptográfico e métodos de compressão. Deixamos os detalhes desses algoritmos para os livros dedicados ao PGP. Quando Alice envia um e-mail para Bob, ela define o algoritmo que ela usou para cada finalidade.

Certificados no PGP e modelo de confiança

O PGP, como outros protocolos que vimos até agora, usa certificados para autenticar chaves públicas. No entanto, o processo é totalmente diferente, conforme explicado a seguir.

Certificados PGP No PGP, não há necessidade de uma Autoridade de Certificação (AC); qualquer pessoa no molho de chaves pode assinar um certificado para qualquer outra pessoa no molho de chaves. Bob pode assinar um certificado para Ted, João, Ana, e assim por diante. Não existe uma hierarquia de confiança no PGP; não há uma árvore de confiança. A falta de estrutura hierárquica pode levar a uma situação na qual Ted pode ter um certificado emitido por Bob e outro certificado emitido por Lisa. Se Alice quiser seguir a cadeia de certificados emitidos para Ted, existem dois caminhos: um se inicia com Bob e outro se inicia com Lisa. Um ponto interessante é que Alice pode confiar plenamente em Bob, porém confiar apenas parcialmente em Lisa. Pode haver múltiplos caminhos na cadeia de confiança partindo de uma autoridade total ou parcialmente confiável até chegar a um certificado. No PGP, o emissor de um certificado é geralmente chamado *introdutor*.

No PGP, pode haver múltiplos caminhos partindo de autoridades total ou parcialmente confiáveis até uma entidade qualquer.

- **Confiança e legitimidade.** Toda a operação do PGP é baseada na confiança no introdutor, na confiança no certificado e na aceitação da legitimidade das chaves públicas.
- **Níveis de confiança dos introdutores.** Com a ausência de uma autoridade central, é óbvio que o molho de chaves não pode ser muito grande se cada usuário tiver que confiar plenamente em todos os outros. Mesmo na vida real, não podemos confiar plenamente em todos que conhecemos. Para resolver esse problema, o PGP permite diferentes níveis de confiança. O número de níveis depende, principalmente, da implementação, mas para simplificar, atribuiremos três níveis de confiança para qualquer introdutor: *nenhum*, *parcial* e *total*. O nível de confiança do introdutor define os níveis de confiança atribuídos pelo usuário a outras pessoas no anel. Por exemplo, Alice pode confiar totalmente em Bob, confiar parcialmente em Ana, e não confiar em João. Não existe um mecanismo no PGP que define como tomar uma decisão sobre a confiabilidade do introdutor; cabe ao usuário tomar essa decisão.
- **Níveis de confiança dos certificados.** Quando Alice recebe um certificado de um introdutor, ela armazena o certificado sob o nome de seu dono (entidade certificada). Ela atribui um nível de confiança a esse certificado. O nível de confiança do certificado é normalmente igual ao nível de confiança atribuído ao introdutor que emitiu o certificado. Considere que Alice confia totalmente em Bob, parcialmente em Ana e Janete e não confia em João. Os seguintes cenários podem ocorrer.

  1. Bob emite dois certificados, um para Linda (com a chave pública K1) e um para Leslie (com a chave pública K2). Alice armazena a chave pública e o certificado de Linda com o nome de Linda e atribui um nível de confiança *total* a esse certificado. Alice também armazena o certificado e a chave pública de Leslie com o nome de Leslie e atribui um nível de confiança total a ele.
  2. Ana emite um certificado para João (com a chave pública K3). Alice armazena o certificado e a chave pública sob o nome de João, mas atribui um nível de confiança *parcial* a ele.
  3. Janete emite dois certificados, um para João (com a chave pública K3) e um para Lee (com a chave pública K4). Alice armazena o certificado de João sob o nome dele e o certificado de Lee sob o nome dele, cada um com um nível de confiança *parcial*. Perceba que João tem agora dois certificados, um emitido por Ana e outro por Janete, cada um com um nível de confiança *parcial*.

4. João emite um certificado para Lisa. Alice pode descartar esse certificado, ou então salvá-lo com um nível de confiança atribuído como *nenhum*.

- **Legitimidade da chave.** O objetivo de usar um nível de confiança para o introdutor e para o certificado é determinar a legitimidade de uma chave pública. Alice precisa saber quão legítimas são as chaves públicas de Bob, João, Lisa, Ana, e assim por diante. O PGP define um procedimento muito claro para determinar a legitimidade da chave. O nível de legitimidade da chave para um usuário corresponde ao nível de confiança ponderado daquele usuário. Por exemplo, considere que atribuímos os seguintes pesos para os níveis de confiança dos certificados:

1. Um peso de 0 para certificados não confiáveis.
2. Um peso de 1/2 para certificados com confiança parcial.
3. Um peso de 1 para certificados com confiança total.

Portanto, para confiar totalmente em uma entidade, Alice precisa de um certificado totalmente confiável ou de dois certificados parcialmente confiáveis. Por exemplo, Alice pode usar a chave pública de João no cenário anterior, pois Ana e Janete emitiram um certificado para João e cada um deles apresenta um nível de confiança de 1/2. Note que a legitimidade de uma chave pública pertencente a uma entidade não tem qualquer relação com o nível de confiança atribuído àquela pessoa. Embora Bob possa usar a chave pública de João para enviar uma mensagem para ele, Alice não pode aceitar qualquer certificado emitido por João porque, para Alice, o nível de confiança de João é *nenhum*.

Modelo confiança no PGP Conforme proposto por Zimmermann, podemos criar um modelo de confiança para qualquer usuário em um molho de chaves tendo o usuário como entidade central. Tal modelo pode ser parecido com o apresentado na Figura 10.34. A figura mostra o modelo de confiança para Alice em algum momento.

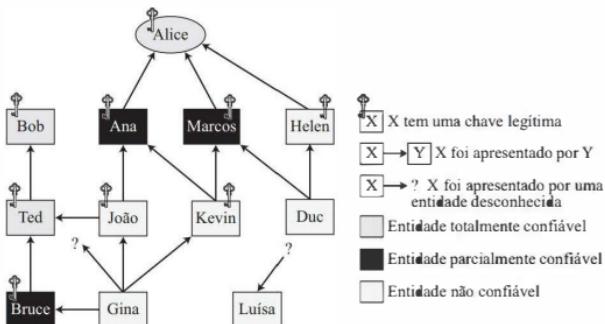



Figura 10.34 Modelo de confiança.

Discutiremos a figura mais a fundo. A Figura 10.34 mostra que existem três entidades no molho de chaves de Alice que têm confiança total (a própria Alice, Bob e Ted). A figura também mostra três entidades com confiança parcial (Ana, Marcos e Bruce). Existem também seis entidades com nenhuma confiança. Nove entidades têm uma chave legítima. Alice pode cifrar mensagens e enviá-las para qualquer uma dessas entidades ou verificar assinaturas recebidas de cada uma dessas

entidades (a chave de Alice nunca é utilizada nesse modelo). Há também três entidades que não têm chaves que sejam consideradas legítimas por Alice.

Bob, Ana e Marcos conseguiram que suas chaves fossem consideradas legítimas enviando-as por e-mail e verificando suas impressões digitais* por telefone. Helen, por outro lado, enviou um certificado emitido por uma AC porque Alice não a considera confiável e a verificação por telefone não é possível. Embora Ted seja totalmente confiável, ele forneceu a Alice um certificado assinado por Bob. João enviou dois certificados a Alice, um assinado por Ted e outro por Ana. Kevin também enviou dois certificados para Alice, um assinado por Ana e outro por Marcos. Cada um desses certificados dá a Kevin meio ponto de legitimidade e, portanto, a chave de Kevin é considerada legítima. Duc enviou dois certificados para Alice, um assinado por Marcos e outro por Helen. Como Marcos é parcialmente confiável e Helen não é confiável, Duc não tem uma chave legítima. Gina enviou quatro certificados, um assinado por uma entidade parcialmente confiável, dois por entidades não confiáveis e um emitido por uma entidade desconhecida. Gina não tem pontos suficientes para que sua chave seja considerada legítima. Luísa enviou um certificado assinado por uma entidade desconhecida. Perceba que Alice pode manter o nome Luísa na tabela caso novos certificados de Luísa apareçam no futuro.

- **Rede de confiança.** O PGP pode, eventualmente , criar uma **rede de confiança** (***web of trust***) entre um grupo de pessoas. Se cada entidade apresentar mais entidades para outras entidades, o molho de chaves públicas de cada entidade ficará cada vez maior e as entidades pertencentes ao anel podem enviar mensagens de e-mail de forma segura umas para as outras.
- **Revogação de chaves.** Pode vir a ser necessário que uma entidade revogue sua chave pública do molho de chaves. Isso pode acontecer se o dono da chave acreditar que a chave tenha sido comprometida (roubada, por exemplo) ou que ela é simplesmente muito antiga para ainda ser considerada segura. Para revogar uma chave, o seu proprietário pode enviar um certificado de revogação assinado por ele próprio. O certificado de revogação deve ser assinado pela chave antiga e disseminado para todas as pessoas no molho de chaves que usam aquela chave pública.

Pacotes PGP

Uma mensagem no PGP consiste em um ou mais pacotes. Durante a evolução do PGP, o formato e o número de tipos de pacotes mudaram. Não discutimos os formatos desses pacotes aqui.

Aplicações do PGP

O PGP tem sido amplamente utilizado para o envio de e-mails pessoais. Ele provavelmente continuará sendo usado com esse propósito.

S/MIME

Outro serviço de segurança projetado para mensagens de correio eletrônico é o **Extensões Seguras/Multifunção para Mensagens de Internet** (S/MIME – Secure/Multipurpose Internet Mail Extension). O protocolo é um aprimoramento do protocolo Extensões Multifunção para Mensagens de Internet (MIME – Multipurpose Internet Mail Extension) que discutimos no Capítulo 2.

Sintaxe para Mensagens Cifradas

Para definir como serviços de segurança, tais como confidencialidade e integridade, podem ser adicionados aos tipos de conteúdo do MIME, o S/MIME especificou a **Sintaxe para Mensagens**

* N. de T.: No PGP, a impressão digital de uma chave corresponde a um resumo criptográfico da chave pública.

Cifradas (CMS – Cryptographic Message Syntax). Essa sintaxe define, em cada caso, o esquema de codificação exato para cada tipo de conteúdo. A seguir, descrevemos os tipos de mensagens e os diferentes subtipos que são criados a partir dessas mensagens. Para mais detalhes, recomenda-se a leitura das RFCs 3369 e 3370.

Tipo de conteúdo: *data* Consiste em uma sequência arbitrária de caracteres. O objeto criado é denominado Data (dados).

Tipo de conteúdo: *signed-data* Fornece apenas integridade dos dados. Ele contém qualquer tipo de dados acompanhados de zero ou mais valores de assinatura. O resultado codificado é um objeto denominado signedData (dados assinados). A Figura 10.35 mostra o processo de criação de um objeto desse tipo. As etapas do processo são dadas a seguir:

1. Para cada signatário, um resumo criptográfico da mensagem é criado a partir do conteúdo, usando o algoritmo de *hash* específico escolhido por aquele signatário.
2. Cada resumo criptográfico da mensagem é assinado usando a chave privada do signatário.
3. O conteúdo, os valores das assinaturas, os certificados e os algoritmos são então combinados para criar o objeto signedData.

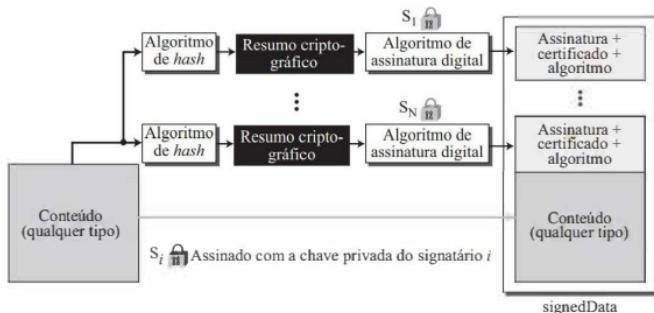



Figura 10.35 Tipo de conteúdo: *signed-data* (dados assinados).

Perceba que, nesse caso, o conteúdo não é necessariamente uma mensagem pessoal. O conteúdo pode ser um documento cuja integridade deve ser preservada. O remetente pode coletar as assinaturas e então enviá-las (ou armazená-las) juntamente com a mensagem.

Tipo de conteúdo: *enveloped-data* É usado para fornecer confidencialidade à mensagem. Ele contém qualquer tipo de mensagem juntamente com zero ou mais chaves cifradas e certificados. O resultado codificado é um objeto denominado envelopedData (dados encapsulados). A Figura 10.36 mostra o processo de criação de um objeto desse tipo.

1. Uma chave de sessão pseudoaleatória é criada para os algoritmos de chave simétrica a serem utilizados.
2. Para cada destinatário, uma cópia da chave de sessão é cifrada com a chave pública daquele destinatário.
3. O conteúdo é cifrado usando o algoritmo especificado e a chave de sessão criada.
4. Os conteúdos cifrados, as chaves de sessão cifradas, o algoritmo utilizado e os certificados são codificados usando Base64.

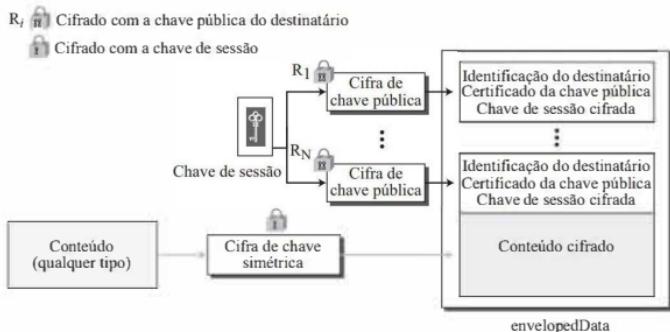



Figura 10.36 Tipo de conteúdo: *enveloped-data* (dados encapsulados).

Perceba que, nesse caso, podemos ter um ou mais destinatários.

Tipo de conteúdo: *digested-data* É usado para fornecer integridade à mensagem. O resultado é normalmente utilizado como o conteúdo para o tipo *enveloped-data*. O resultado codificado é um objeto denominado digestedData (dados de resumo criptográfico). A Figura 10.37 mostra o processo de criação de um objeto deste tipo.

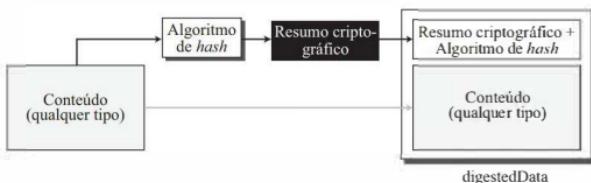



Figura 10.37 Tipo de conteúdo: *digested-data* (dados de resumo criptográfico).

1. Um resumo criptográfico da mensagem é calculado a partir do conteúdo.
2. O resumo criptográfico da mensagem, o algoritmo e o conteúdo são agrupados para criar o objeto denominado digestedData.

Tipo de conteúdo: *encrypted-data* É usado para criar uma versão cifrada de qualquer tipo de conteúdo. Embora se pareça com o tipo de conteúdo *enveloped-data*, o tipo de conteúdo denominado *encrypted-data* (dados cifrados) não tem destinatário. Ele pode ser usado para armazenar os dados cifrados em vez de transmiti-los. O processo é muito simples: o usuário utiliza qualquer chave (normalmente derivada de uma senha) e qualquer algoritmo para cifrar o conteúdo. O conteúdo cifrado é armazenado sem incluir a chave ou o algoritmo. O objeto criado é denominado encryptedData (dados cifrados).

Tipo de conteúdo: *authenticated-data* É usado para fornecer autenticação dos dados. O objeto é denominado authenticatedData (dados autenticados). A Figura 10.38 ilustra o processo.

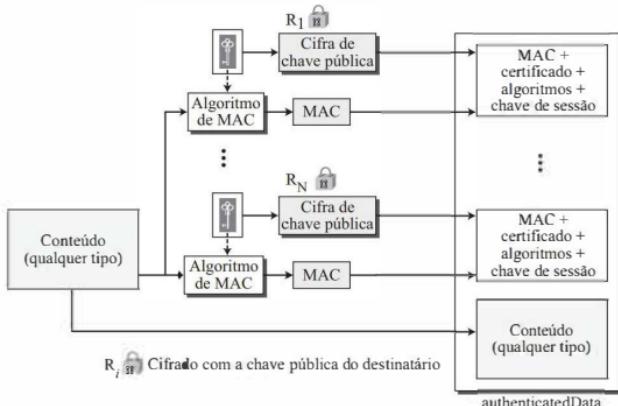



Figura 10.38 Tipo de conteúdo: *authenticateddata* (dados autenticados).

1. Usando um gerador pseudoaleatório, é gerada uma chave de MAC para cada destinatário.
2. A chave de MAC é cifrada usando a chave pública do destinatário.
3. Um MAC é criado para o conteúdo.
4. O conteúdo, o MAC, os algoritmos e outras informações são agrupados para formar o objeto denominado authenticatedData.

Gerenciamento de chaves

O gerenciamento de chaves no S/MIME é uma combinação do gerenciamento de chaves utilizada pelo X.509 e pelo PGP. O S/MIME usa certificados de chave pública assinados pelas autoridades certificadoras definidas pelo X.509. No entanto, o usuário é responsável por manter a rede de confiança para verificar as assinaturas conforme especificado pelo PGP.

Algoritmos criptográficos

O S/MIME aceita diversos algoritmos criptográficos. Deixamos os detalhes desses algoritmos para os livros dedicados à segurança na Internet.

Exemplo 10.10

O exemplo a seguir mostra um objeto do tipo *enveloped-data* (dados encapsulados) no qual uma mensagem curta é cifrada usando o AES.

```
Content-Type: application/pkcs7-mime; mime-type=enveloped-data
Content-Transfer-Encoding: Base64
Content-Description: attachment
name="relatório.txt";
cb32ut67f4bhijHU21oi87eryb0287hmnklsgFDoY8bc659GhIGfH6543mhjkdsaH23YjBnmN
ybmlkzjhgfdyhGe23Kjk34XiuD678Es16se09jy76jHuytTMDcbnmlkjgfFdiuyu678543m0n3hG
34un12P2454Hoi87e2ryb0H2MjN6KuyrlsgFDoY897fk923jljk1301XiuD6gh78EsUyT23y
```

Aplicações do S/MIME

Prevê-se que o S/MIME será a escolha da indústria para fornecer segurança para mensagens de e-mail comerciais.

10.4.2 Segurança na camada de transporte

Dois protocolos são dominantes atualmente para fornecer segurança na camada de transporte: o protocolo **Camada de Sockets Segura** (SSL – Secure Sockets Layer) e o protocolo **Segurança da Camada de Transporte** (TLS – Transport Layer Security). Este último é, na verdade uma versão criada pela IETF do SSL. Discutimos o SSL nesta seção; o TLS é muito semelhante. A Figura 10.39 mostra a posição do SSL e do TLS no modelo Internet.

Figura 10.39 Localização do SSL e do TLS no modelo Internet.

Um dos objetivos desses protocolos é fornecer autenticação de servidores e de clientes, confidencialidade dos dados e integridade dos dados. Programas cliente-servidor da camada de aplicação que usam os serviços do TCP, como o HTTP (ver Capítulo 2), podem encapsular seus dados em pacotes SSL (HTTPS). Se o servidor e o cliente são capazes de executar programas SSL (ou TLS), então o cliente pode usar o URL *https://...* em vez de *http://...* para permitir que mensagens HTTP sejam encapsuladas em pacotes SSL (ou TLS). Por exemplo, números de cartão de crédito podem ser transferidos com segurança via Internet por compradores conectados à rede.

Arquitetura do SSL

O SSL foi projetado para fornecer serviços de segurança e de compressão para dados gerados pela camada de aplicação. Tipicamente , o SSL pode receber dados de qualquer protocolo de camada de aplicação, mas, em geral, o protocolo é o HTTP. Os dados recebidos da aplicação são compactados (operação opcional), assinados e cifrados. Os dados são, então, passados para um protocolo confiável da camada de transporte, como o TCP. A Netscape desenvolveu o SSL em 1994. As versões 2 e 3 foram lançadas em 1995. Nesta seção, discutimos o SSLv3.

Serviços

O SSL fornece diversos serviços para os dados recebidos da camada de aplicação.

- **Fragmentação.** Primeiramente, o SSL divide os dados em blocos de 2^{14} *bytes* ou menos.
- **Compressão.** Cada fragmento de dados é compactado usando um dos métodos de compressão sem perdas negociados entre o cliente e o servidor. Esse serviço é opcional.
- **Integridade da mensagem.** Para preservar a integridade dos dados, o SSL usa uma função de *hash* com chave para criar um MAC.

- **Confidencialidade.** Para garantir a confidencialidade, os dados originais e o MAC são cifrados usando criptografia de chave simétrica.
- **Enquadramento.** Um cabeçalho é adicionado à carga útil cifrada. A carga útil é então encaminhada para um protocolo confiável da camada de transporte.

Algoritmos de troca de chave

Para transmitir uma mensagem autenticada e confidencial, o cliente e o servidor precisam cada um de um conjunto de segredos criptográficos. Entretanto, para criar esses segredos, um segredo pré-mestre deve ser estabelecido entre as duas partes. O SSL define diversos métodos de troca de chaves para estabelecer esse segredo pré-mestre.

Algoritmos de cifração/decifração

O cliente e o servidor também precisam concordar com um conjunto de algoritmos de cifração e decifração.

Algoritmos de *hash*

O SSL usa algoritmos de *hash* com chave para garantir a integridade das mensagens (autenticação de mensagens). Diversos algoritmos podem ser usados para essa finalidade.

Conjunto de algoritmos criptográficos

A combinação de algoritmos de troca de chaves, de geração de MAC e de cifração/decifração define um **conjunto de algoritmos criptográficos** (*cipher suite*) para cada sessão SSL.

Algoritmos de compressão

A compressão é opcional no SSL. Nenhum algoritmo de compressão específico é definido. Portanto, um sistema pode usar qualquer algoritmo de compressão que desejar.

Geração de parâmetros criptográficos

Para garantir a integridade e a confidencialidade das mensagens, o SSL requer seis elementos criptográficos: quatro chaves e dois IVs (*Initialization Vectors*, ou Vetores de Inicialização). O cliente precisa de uma chave para a autenticação de mensagens, outra para a cifração e um IV usado no início do processo criptográfico. O servidor precisa das mesmas informações. O SSL exige que as chaves para a comunicação em uma direção sejam diferentes daquelas usadas para a comunicação na outra direção. Se houver um ataque em uma direção, a outra direção não é afetada. Os parâmetros são gerados usando o seguinte procedimento:

1. O cliente e o servidor trocam dois números aleatórios; um é criado pelo cliente e o outro pelo servidor.
2. O cliente e o servidor trocam um *segredo pré-mestre* usando um dos algoritmos de troca de chaves predefinidos.
3. Um *segredo mestre* de 48 *bytes* é criado a partir do segredo pré-mestre por meio da aplicação de duas funções de *hash* (SHA-1 e MD5), conforme mostra a Figura 10.40.
4. O segredo mestre é usado para criar *material criptográfico* de comprimento variável por meio da aplicação do mesmo conjunto de funções de *hash* e da concatenação de constantes diferentes, conforme mostra a Figura 10.41. O módulo é repetido até que um comprimento adequado de material criptográfico seja criado.

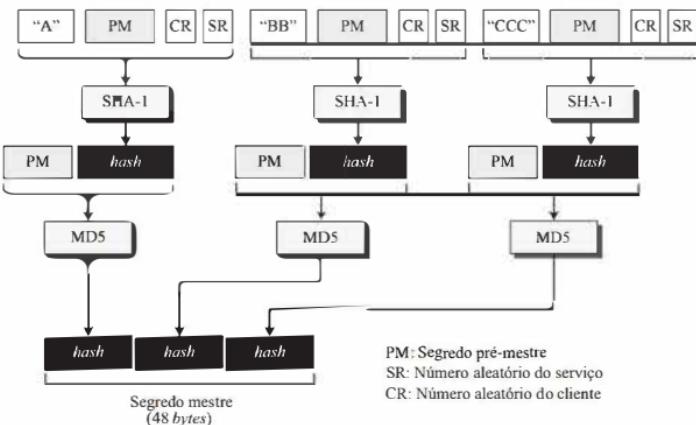



Figura 10.40 Cálculo do segredo mestre a partir do segredo pré-mestre.

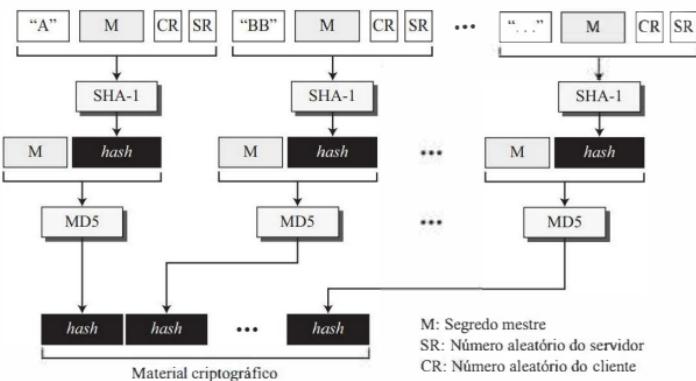



Figura 10.41 Cálculo do material criptográfico a partir do segredo mestre.

Note que o comprimento do bloco de material criptográfico depende do conjunto de algoritmos criptográficos escolhidos e do tamanho das chaves necessárias por esse conjunto.

5. Seis segredos diferentes são extraídos do material criptográfico, conforme mostra a Figura 10.42.

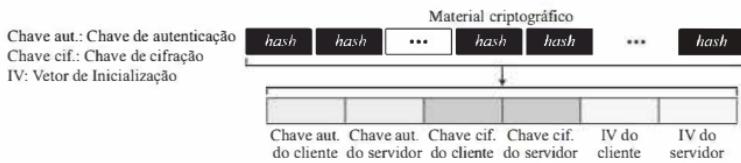



Figura 10.42 Extrações de segredos criptográficos a partir do material criptográfico.

Sessões e conexões

O SSL faz distinção entre uma *conexão* e uma *sessão*. Uma sessão é uma associação entre um cliente e um servidor. Depois que uma sessão é estabelecida, as duas partes possuem informações em comum, tais como o identificador da sessão, o certificado de autenticação de cada um deles (se necessário), o método de compressão (se necessário), o conjunto de algoritmos criptográficos e um segredo mestre que é usado para criar chaves para a cifração e a autenticação de mensagens.

Para que duas entidades troquem dados, o estabelecimento de uma sessão é necessário, porém não é suficiente; as entidades precisam criar uma conexão entre elas. As duas entidades trocam dois números aleatórios entre elas e criam, usando o segredo mestre, as chaves e os parâmetros necessários para a troca de mensagens envolvendo autenticação e confidencialidade.

Uma sessão pode envolver diversas conexões. Uma conexão entre duas partes pode ser encerrada e restabelecida na mesma sessão. Quando uma conexão é encerrada, as duas partes também podem encerrar a sessão, mas isso não é obrigatório. Uma sessão pode ser suspensa e reiniciada.

Quatro protocolos

Discutimos a ideia por trás do SSL sem mostrar como o SSL exerce suas funções. O SSL define quatro protocolos em duas camadas, conforme mostra a Figura 10.43.

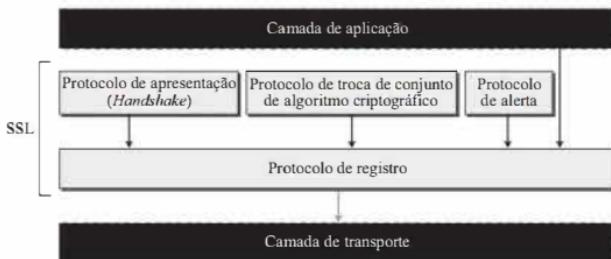



Figura 10.43 Quatro protocolos SSL.

O Protocolo de Registro atua como transportador. Ele transporta mensagens provenientes dos três outros protocolos, bem como os dados provenientes da camada de aplicação. As mensagens vindas do Protocolo de Registro são as cargas úteis do protocolo da camada de transporte, normalmente o TCP. O Protocolo de Apresentação (também conhecido como protocolo de *Handshake* ou

de "Aperto de Mãos") fornece parâmetros de segurança para o Protocolo de Registro. Ele estabelece um conjunto de algoritmos e fornece as chaves e parâmetros de segurança correspondentes. Ele também autentica o servidor em relação ao cliente e o cliente em relação ao servidor, se necessário. O Protocolo de Troca de Conjunto de Algoritmos Criptográficos (também chamado *ChangeCipherSpec*) é usado para sinalizar se os segredos criptográficos estão prontos. O Protocolo de Alerta é usado para reportar condições anormais. Discutiremos brevemente esses protocolos nesta seção.

Protocolo de apresentação

O **Protocolo de Apresentação** (ou Protocolo de *Handshake*) usa mensagens para negociar o conjunto de algoritmos criptográficos, para autenticar o servidor em relação ao cliente e o cliente em relação ao servidor, se necessário, bem como para trocar informações usadas na construção dos segredos criptográficos. A apresentação é feita em quatro fases, conforme mostra a Figura 10.44.

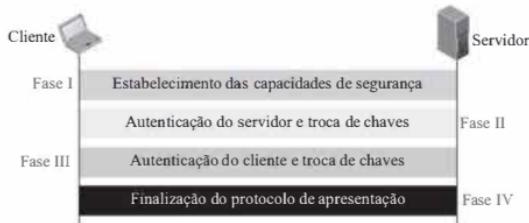



Figura 10.44 Protocolo de Apresentação (*Handshake*).

Fase I: Estabelecimento das capacidades de segurança Na Fase I, o cliente e o servidor anunciam suas capacidades de segurança (os algoritmos suportados por eles) e escolhem aqueles que sejam convenientes para ambos. Nessa fase, o identificador da sessão é estabelecido e o conjunto de algoritmos criptográficos é escolhido. As partes concordam com um método de compressão em particular. Finalmente, dois números aleatórios são gerados, um pelo cliente e outro pelo servidor, para serem usados na criação de um segredo mestre, como vimos anteriormente. Depois da Fase I, o cliente e o servidor sabem a versão do SSL, os algoritmos de criptografia, o método de compressão e os dois números aleatórios para a geração de chaves.

Fase II: Autenticação do servidor e troca de chaves Na Fase II, o servidor se autentica, se necessário. O servidor pode enviar o seu certificado, sua chave pública, e também pode solicitar certificados do cliente. Depois da Fase II, o servidor encontra-se autenticado em relação ao cliente, e o cliente sabe a chave pública do servidor, se necessário.

Fase III: Autenticação do cliente e troca de chaves A Fase III foi concebida para autenticar o cliente. Depois da Fase III, o cliente encontra-se autenticado em relação ao servidor e tanto o cliente como o servidor conhecem o segredo pré-mestre.

Fase IV: Finalização e encerramento Na Fase IV, o cliente e o servidor enviam mensagens para alterar os algoritmos criptográficos especificados e para finalizar o Protocolo de Apresentação.

Protocolo de Troca de Conjunto de Algoritmos Criptográficos

Vimos que a negociação do conjunto de algoritmos criptográficos e a geração de segredos criptográficos são realizadas gradualmente durante o Protocolo de Apresentação. A questão agora é: quando as duas partes usam esses parâmetros e segredos? O SSL dita que as partes não podem usar esses parâmetros ou segredos até terem enviado ou recebido uma mensagem especial, denominada

mensagem de troca do conjunto de algoritmos criptográficos (*ChangeCipherSpec*), que é transmitida durante o Protocolo Apresentação e define o *Protocolo de Troca de Conjunto de Algoritmos Criptográficos*. Não se trata apenas de enviar ou receber uma mensagem. O servidor e o cliente precisam de dois estados, não apenas de um. Um estado, o estado pendente, mantém o registro dos parâmetros e segredos. O outro estado, o estado ativo, guarda os parâmetros e segredos usados pelo Protocolo de Registro para assinar/verificar e cifrar/decifrar mensagens. Além disso, cada estado possui dois conjuntos de valores: *leitura* (entrada) e *escrita* (saída).

Protocolo de alerta

O SSL usa o *Protocolo de Alerta* para relatar erros e condições anormais. Ele usa apenas uma mensagem que descreve o problema e o seu nível de gravidade (aviso ou fatal).

Protocolo de registro

O *Protocolo de Registro* transmite mensagens vindas da camada superior (Protocolo de Apresentação, Protocolo de Troca de Conjunto de Algoritmos Criptográficos, Protocolo de Alerta, ou camada de aplicação). A mensagem é fragmentada e, opcionalmente, compactada; um MAC é adicionado à mensagem compactada usando o algoritmo de *hash* negociado. O fragmento compactado e o MAC são cifrados usando a cifra negociada. Finalmente, o cabeçalho SSL é adicionado à mensagem cifrada. A Figura 10.45 ilustra esse processo no lado do cliente. O processo no lado do servidor é o inverso.

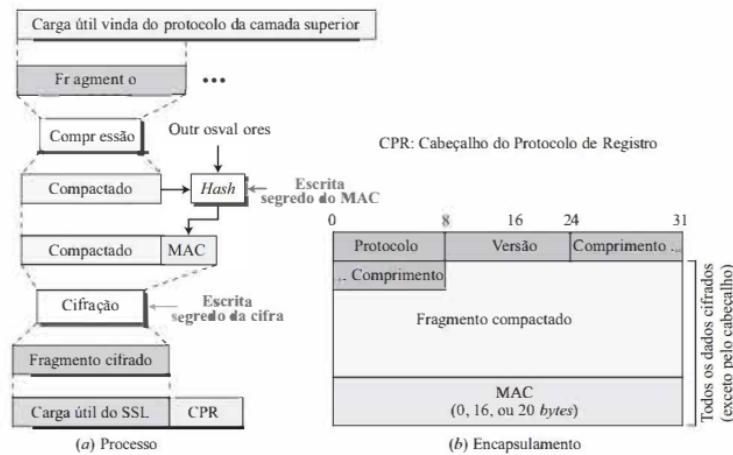



Figura 10.45 Processamento realizado pelo Protocolo de Registro.

10.4.3 Segurança na camada de rede

Começamos esta seção com uma discussão sobre a segurança na camada de rede. Embora tenhamos discutido segurança na camada de aplicação e na camada de transporte nas seções anteriores, também precisamos de segurança na camada de rede por três razões. Primeiro, nem todos os programas cliente-servidor são protegidos na camada de aplicação. Segundo, nem todos os programas cliente-servidor na camada de aplicação utilizam os serviços do TCP para poderem ser protegidos pelo

SSL ou TLS que discutimos para a camada de transporte; alguns programas utilizam os serviços do UDP. Terceiro, muitas aplicações, como protocolos de roteamento, usam diretamente os serviços do IP; elas precisam de serviços de segurança na camada IP.

A **Segurança IP** (IPSec – IP Security) consiste em uma coleção de protocolos projetados pela Força-Tarefa de Engenharia da Internet (IETF – Internet Engineering Task Force) para fornecer segurança para pacotes na camada de rede. O IPSec ajuda a criar pacotes autenticados e confidenciais na camada IP.

Dois modos

O IPSec opera em um de dois modos distintos: modo de transporte ou modo túnel.

Modo de transporte

No **modo de transporte**, o IPSec protege os dados entregues pela camada de transporte para a camada de rede. Em outras palavras, o modo de transporte protege a carga útil a ser encapsulada na camada de rede, conforme mostra a Figura 10.46.

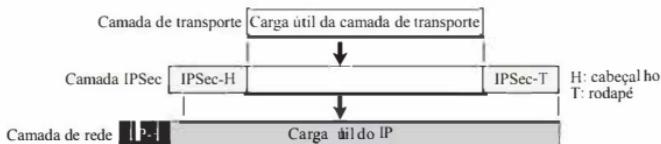



Figura 10.46 IPSec no modo de transporte.

Observe que o modo de transporte não protege o cabeçalho IP. Ou seja, o modo de transporte não protege todo o pacote IP; ele protege somente a parte do pacote proveniente da camada de transporte (a carga útil da camada IP). Nesse modo, o cabeçalho (e o rodapé) IPSec são adicionados à informação proveniente da camada de transporte. O cabeçalho IP é adicionado mais tarde.

O IPSec no modo de transporte não protege o cabeçalho IP; ele protege apenas a carga útil proveniente da camada de transporte.

O modo de transporte é normalmente utilizado quando precisamos de proteção de dados *host* a *host* (fim a fim). O *host* emissor usa o IPSec para autenticar e/ou cifrar a carga útil entregue pela camada de transporte. O *host* receptor usa o IPSec para verificar a autenticidade do pacote IP e/ou decifrá-lo antes de entregá-lo à camada de transporte. A Figura 10.47 ilustra esse conceito.

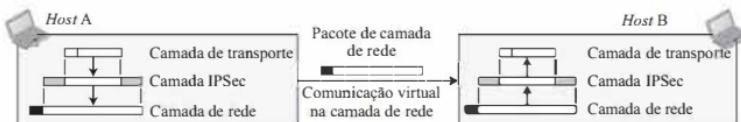



Figura 10.47 Modo de transporte em ação.

Modo túnel

No **modo túnel**, o IPSec protege o pacote inteiro. Ele recebe um pacote IP, incluindo o cabeçalho, aplica os métodos de segurança do IPSec ao pacote todo e então adiciona um novo cabeçalho IP, conforme mostra a Figura 10.48.

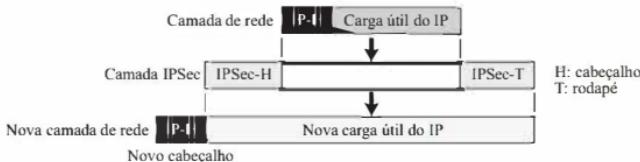



Figura 10.48 IPSec no modo túnel.

O novo cabeçalho IP, conforme veremos em breve, carrega informações diferentes daquelas presentes no cabeçalho IP original. O modo túnel é normalmente usado entre dois roteadores, entre uma estação e um roteador ou entre um roteador e uma estação, conforme mostra a Figura 10.49. O pacote original inteiro é protegido contra atacantes no caminho entre o emissor e o receptor, como se o pacote inteiro passasse por um túnel imaginário.

Figura 10.49 Modo túnel em ação.

O IPSec no modo túnel protege o cabeçalho IP original.

Comparação

No modo de transporte, a camada IPSec fica entre a camada de transporte e a camada de rede. No modo túnel, o fluxo vai da camada de rede para a camada IPSec e então de volta para a camada de rede novamente. A Figura 10.50 compara os dois modos.

Dois protocolos de segurança

O IPSec define dois protocolos – o Protocolo AH (*Authentication Header*, ou Cabeçalho de Autenticação) e o Protocolo ESP (*Encapsulating Security Payload*, ou Encapsulamento de Dados de Segurança) – para fornecer autenticação e/ou cifração de pacotes no nível do IP.

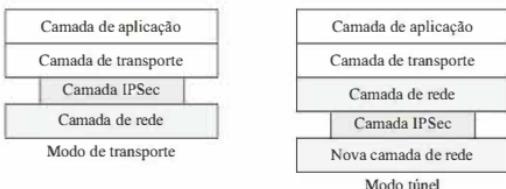



Figura 10.50 Modo de transporte *versus* modo túnel.

Cabeçalho de Autenticação

O **protocolo AH** (*Authentication Header*, ou Cabeçalho de Autenticação) foi projetado para autenticar o *host* de origem e para garantir a integridade da carga útil transportada pelo pacote IP. O protocolo usa uma função de *hash* e uma chave simétrica (secreta) para criar um resumo criptográfico da mensagem; o resumo criptográfico é inserido no cabeçalho de autenticação (ver seção sobre MAC). O AH é então colocado no local apropriado, com base no modo do IPSec (transporte ou túnel). A Figura 10.51 mostra os campos e a posição do cabeçalho de autenticação no modo de transporte.

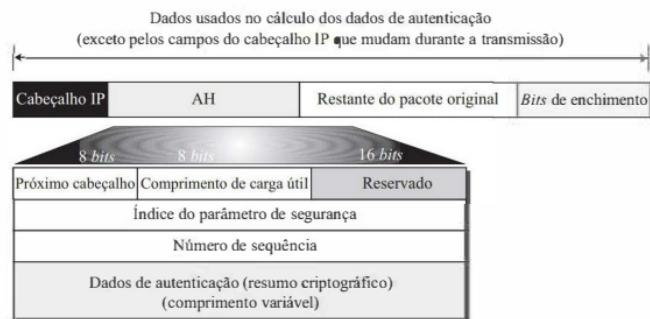



Figura 10.51 Protocolo de Cabeçalho de Autenticação (AH).

Quando um datagrama IP carrega um cabeçalho de autenticação, o valor original do campo de protocolo no cabeçalho IP é substituído pelo valor 51. Um campo no interior do cabeçalho de autenticação (o campo de *próximo cabeçalho*) carrega o valor original do campo de protocolo (o tipo de carga útil sendo transportada pelo datagrama IP). O processo de adição de um cabeçalho de autenticação segue os seguintes passos:

1. Um cabeçalho de autenticação é adicionado à carga útil com o campo de dados de autenticação fixado em 0.
2. *Bits* de enchimento (*padding*) podem ser adicionados para se obter o comprimento total apropriado para uma função de *hash* em particular.

3. A função de *hash* é aplicada ao pacote inteiro. No entanto, apenas os campos do cabeçalho IP, que não se alteram durante a transmissão, são incluídos no cálculo do resumo criptográfico da mensagem (dados de autenticação).
4. Os dados de autenticação são inseridos no cabeçalho de autenticação.
5. O cabeçalho IP é adicionado após o valor do seu campo de protocolo ser alterado para 51.

A seguir, fornecemos uma breve descrição de cada campo:

- **Próximo cabeçalho.** O campo de próximo cabeçalho, de 8 *bits*, define o tipo de carga transportada pelo datagrama IP (por exemplo, TCP, UDP, ICMP ou OSPF).
- **Comprimento da carga útil.** O nome deste campo de 8 *bits* engana. Ele não especifica o comprimento da carga útil; o que ele especifica é o comprimento do cabeçalho de autenticação em múltiplos de 4 *bytes*, mas sem incluir os primeiros 8 *bytes*.
- **Índice do parâmetro de segurança.** O Índice do Parâmetro de Segurança (SPI – Security Parameter Index) é um campo de 32 *bits* que desempenha o papel de um identificador de circuito virtual. Ele é idêntico para todos os pacotes enviados durante uma conexão denominada Associação de Segurança (discutida mais adiante).
- **Número de sequência.** Um número de sequência de 32 *bits* que fornece informações sobre ordenação para uma sequência de datagramas. Os números de sequência previnem ataques de repetição. Perceba que o número de sequência não é repetido mesmo se um pacote for retransmitido. Um número de sequência não é reiniciado após atingir o valor de 2^{32}; uma nova conexão deve ser estabelecida após isso acontecer.
- **Dados de autenticação.** Finalmente, o campo de dados de autenticação é o resultado da aplicação de uma função de *hash* sobre o datagrama IP inteiro, exceto pelos campos que são alterados em trânsito (por exemplo, o campo de TTL).

O protocolo AH fornece serviços de autenticação da origem e integridade dos dados, porém não de confidencialidade.

Encapsulamento de Dados de Segurança

O protocolo AH não fornece confidencialidade, apenas autenticação da origem e integridade dos dados. Após o protocolo AH ter sido criado, o IPSec definiu um protocolo alternativo, denominado **Encapsulamento de Dados de Segurança** (ESP – Encapsulating Security Payload), que fornece os serviços de autenticação de origem, integridade e confidencialidade. O ESP acrescenta um cabeçalho e um rodapé ao pacote. Os dados de autenticação do ESP são inseridos no final do pacote, o que facilita o seu cálculo. A Figura 10.52 mostra a localização do cabeçalho e do rodapé no ESP.

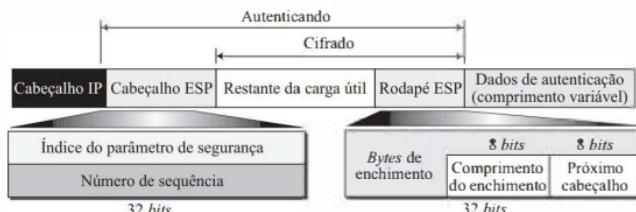



Figura 10.52 Encapsulamento de Dados de Segurança (ESP).

Quando um datagrama IP transporta um cabeçalho e um rodapé ESP, o valor do campo de protocolo no cabeçalho IP é 50. Um campo no interior do rodapé ESP (o campo de próximo cabeçalho) guarda o valor original do campo de protocolo (o tipo de carga útil sendo transportada pelo datagrama IP, como TCP ou UDP). O procedimento empregado pelo protocolo ESP consiste nos seguintes passos:

1. Um rodapé ESP é adicionado à carga útil.
2. A carga útil e o rodapé são cifrados.
3. O cabeçalho ESP é adicionado.
4. O cabeçalho ESP, a carga útil e o rodapé ESP são usados para criar os dados de autenticação.
5. Os dados de autenticação são inseridos no final do rodapé ESP.
6. O cabeçalho IP é adicionado depois que o valor do campo de protocolo é alterado para 50.

Os campos de cabeçalho e rodapé são os seguintes:

- **Índice do parâmetro de segurança.** O campo de índice do parâmetro de segurança, de 32 *bits*, é semelhante ao campo de mesmo nome definido no protocolo AH.
- **Número de sequência.** O campo de número de sequência, de 32 *bits*, é semelhante ao campo de mesmo nome definido no protocolo AH.
- ***Bytes* de enchimento.** Este campo de comprimento variável (0 a 255 *bytes*) repleto de 0s serve como enchimento (*padding*)
- **Comprimento do enchimento.** O campo de comprimento do enchimento, de 8 *bits*, define o número de *bytes* de enchimento. O valor do campo fica entre 0 e 255; raramente o valor máximo é utilizado.
- **Próximo cabeçalho.** O campo de próximo cabeçalho, de 8 *bits*, é semelhante àquele de mesmo nome definido no protocolo AH. Ela serve ao mesmo propósito que o campo de protocolo no cabeçalho IP antes do encapsulamento.
- **Dados de autenticação.** Finalmente, o campo de dados de autenticação é o resultado da aplicação de um esquema de autenticação a partes do datagrama. Perceba a diferença entre os dados de autenticação no AH e no ESP. No AH, uma parte do cabeçalho IP é incluída no cálculo dos dados de autenticação; no ESP, isso não acontece.

IPv4 e IPv6

O IPSec suporta tanto IPv4 como IPv6. No IPv6, entretanto, o AH e o ESP fazem parte do cabeçalho de extensão.

AH *versus* ESP

O protocolo ESP foi projetado depois que o protocolo AH já estava em uso. O ESP faz tudo o que o AH faz com funcionalidades adicionais (confidencialidade). O AH, na realidade, não é necessário. No entanto, a implementação do AH já se encontra disponível em alguns produtos comerciais, o que significa que o AH continuará a fazer parte da Internet até que esses produtos saiam de linha.

Serviços fornecidos pelo IPSec

Os dois protocolos, AH e ESP, podem oferecer diversos serviços de segurança para pacotes na camada de rede. A Tabela 10.1 mostra a lista de serviços disponibilizados por cada protocolo.

Tabela 10.1 Serviços do IPSec.

Serviços	AH	ESP
Controle de acesso	Sim	Sim
Autenticação de mensagens (integridade de mensagens)	Sim	Sim
Autenticação de entidades (autenticação da origem dos dados)	Sim	Sim
Confidencialidade	Não	Sim
Proteção contra ataques de repetição	Sim	Sim

Controle de acesso

O IPSec fornece controle de acesso indiretamente, utilizando uma Base de Dados de Associações de Segurança (SAD – Security Association Database), conforme veremos na próxima seção. Quando um pacote chega a um destino e não existe uma Associação de Segurança já estabelecida para ele, o pacote é descartado.

Integridade de mensagens

A integridade da mensagem é preservada tanto no AH como no ESP. Um resumo criptográfico dos dados é criado e enviado pelo emissor de modo que a integridade dos dados possa ser verificada pelo receptor.

Autenticação de entidades

A Associação de Segurança e os dados de autenticação gerados a partir dos dados enviados pelo emissor servem para autenticar esse emissor tanto no AH como no ESP.

Confidencialidade

A cifração da mensagem no ESP fornece confidencialidade; o AH, não. Se o serviço de confidencialidade for necessário, deve-se utilizar o ESP em vez do AH.

Proteção contra ataques de repetição

Em ambos os protocolos, o ataque de repetição é prevenido por meio do uso de números de sequência e de uma janela deslizante no receptor. Cada cabeçalho IPSec contém um número de sequência único quando a Associação de Segurança é estabelecida. O número inicia-se em 0 e é incrementado até que atinja o valor de $2^{32} - 1$. Quando o número de sequência chega ao seu valor máximo, ele é reiniciado em 0 e, ao mesmo tempo, a antiga Associação de Segurança (ver próxima seção) é eliminada e uma nova é estabelecida. Para evitar o processamento de pacotes duplicados, o IPSec exige a utilização de uma janela de tamanho fixo no receptor. O tamanho da janela é determinado pelo receptor, mas seu valor-padrão é 64.

Associação de segurança

A Associação de Segurança é um aspecto muito importante do IPSec. O IPSec requer o estabelecimento de uma relação lógica, denominada **Associação de Segurança** (SA), entre dois *hosts*. Essa seção primeiramente discute a ideia e, em seguida, mostra como ela é usada no IPSec.

Ideia geral da associação de segurança

A Associação de Segurança é um contrato entre duas partes; ela cria um canal seguro entre elas. Vamos supor que Alice precisa se comunicar unidirecionalmente com Bob. Se Alice e Bob estão interessados apenas no aspecto de segurança da confidencialidade, eles podem compartilhar uma chave secreta entre eles. Podemos dizer que há duas ASs entre Alice e Bob, uma AS de saída e uma AS de entrada. Cada uma delas armazena o valor da chave em uma variável e o nome do algoritmo de cifração/decifração na outra. Alice usa o algoritmo e a chave para cifrar uma mensagem para Bob; Bob usa o algoritmo e a chave quando ele precisa decifrar uma mensagem recebida de Alice. A Figura 10.53 mostra uma AS simples.

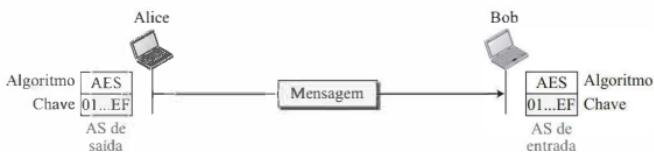



Figura 10.53 AS simples.

As Associações de Segurança podem ser mais complexas se as duas partes precisarem de integridade e autenticação das mensagens. Cada associação requer outros dados, como o algoritmo usado para fornecer integridade à mensagem, a chave, e outros parâmetros. A AS pode ser muito mais complexa se as partes precisarem usar algoritmos específicos e parâmetros específicos para diferentes protocolos, tais como IPSec AH ou IPSec ESP.

Base de Dados de Associações de Segurança

Uma Associação de Segurança pode ser muito complexa. Isto é particularmente verdadeiro se Alice deseja enviar mensagens para muitas pessoas e se Bob precisa receber mensagens de muitas pessoas. Além disso, cada lado da comunicação precisa ter tanto ASs de entrada como ASs de saída para que a comunicação bidirecional seja possível. Em outras palavras, precisamos de um conjunto de ASs, que podem ser agrupadas em uma base de dados. Tal base de dados é denominada **Base de Dados de Associações de Segurança** (SAD – Security Association Database). Essa base de dados pode ser vista como uma tabela bidirecional, na qual cada linha define uma única AS. Normalmente, existem duas SADs, uma de entrada e outra de saída. A Figura 10.54 mostra o conceito de SADs de entrada e de saída para uma entidade.

Índice	NS	IE	JAR	AH/ESP	TV	Modo	MTU
< SPI, ED, P >							
• • •							
< SPI, ED, P >							

Base de Dados de Associação de Segurança

NS: Número de sequência
IE: Indicador de estouro do contador de sequência
JAR: Janela antirrepetição
TV: Tempo de vida
MTU: Unidade Máxima de Transmissão do caminho
SPI: Índice do parâmetro de segurança
ED: Endereço de destino
AH/ESP: Informação
P: Protocolo
Modo: Indicador do modo IPSec

Figura 10.54 Base de Dados de Associações de Segurança (SAD).

Quando um *host* precisa enviar um pacote que deve conter um cabeçalho IPSec, ele deve encontrar a linha correspondente na SAD de saída para determinar a informação usada para aplicar

segurança ao pacote. Analogamente, quando um *host* recebe um pacote contendo um cabeçalho IPSec, o *host* precisa encontrar a linha correspondente na SAD de entrada para determinar as informações necessárias para verificar a segurança do pacote. Essa busca deve ser específica, no sentido que o *host* que recebe o pacote precisa ter certeza de que está usando a informação correta para processar o pacote. Cada linha em uma SAD de entrada é selecionada por meio de um índice triplo: o índice do parâmetro de segurança (um número de 32 *bits* que determina o AS no destino), o endereço de destino e o protocolo (AH ou ESP).

Política de segurança

Outro aspecto importante do IPSec é a **Política de Segurança**, que define o tipo de segurança aplicada a um pacote quando ele é enviado ou quando chega. Antes de usar a SAD, discutida na seção anterior, um *host* deve determinar a política predefinida para o pacote.

Base de Dados de Políticas de Segurança

Cada *host* usando o protocolo IPSec precisa manter uma **Base de Dados de Políticas de Segurança** (SPD – Security Policy Database). Novamente, é necessário haver uma SPD de entrada e uma SPD de saída. Cada linha na SPD pode ser acessada usando um índice sêxtuplo: endereço de origem, endereço de destino, nome, protocolo, porta de origem e porta de destino, conforme mostra a Figura 10.55. O nome geralmente define uma entidade de DNS. O protocolo é o AH ou o ESP.

Índice	Política
< EO, ED, Nome, P, PortaO, PortaD >	
• • •	
< EO, ED, Nome, P, PortaO, PortaD >	

EO: Endereço de origem PortaO: Porta de origem
ED: Endereço de destino PortaD: Porta de destino
P: Protocolo

Figura 10.55 Base de Dados de Políticas de Segurança (SPD).

SPD de saída Quando um pacote deve ser enviado, a SPD de saída é consultada. A Figura 10.56 mostra o processamento de um pacote por um remetente. A entrada para a SPD de saída é o índice sêxtuplo; a saída pode ser uma das três possibilidades a seguir: descartar (o pacote não pode ser enviado), ignorar (ignorar o cabeçalho de segurança) ou aplicar (aplicar os mecanismos de segurança de acordo com o SAD; se não existir uma SAD, criar uma).

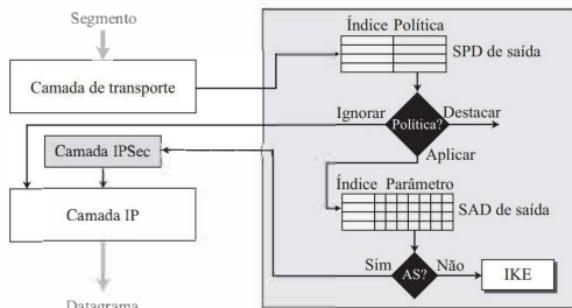



Figura 10.56 Processamento na saída.

SPD de entrada Quando um pacote chega, a SPD de entrada é consultada. Cada linha na SPD de entrada também é acessada usando o mesmo índice sêxtuplo. A Figura 10.57 mostra o processamento de um pacote por um receptor.

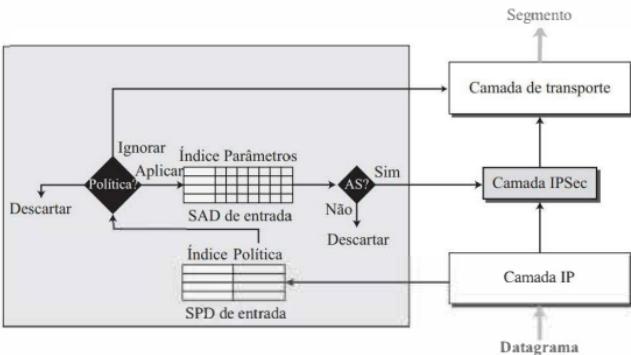



Figura 10.57 Processamento na entrada.

A entrada para a SPD de entrada é o índice sêxtuplo; a saída pode ser uma das três possibilidades a seguir: descartar (descartar o pacote), ignorar (ignorar a segurança e entregar o pacote para a camada de transporte) ou aplicar (aplicar a política usando a SAD).

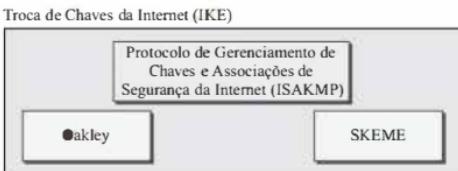
Troca de Chaves da Internet

O **Troca de Chaves da Internet** (IKE – Internet Key Exchange) é um protocolo projetado para criar associações de segurança de entrada e de saída. Conforme discutimos na seção anterior, quando um *host* precisa enviar um pacote IP, ele consulta a Base de Dados de Políticas de Segurança (SPD) para verificar se há uma AS para esse tipo de tráfego. Se não houver uma AS, o IKE é invocado para estabelecer uma.

O IKE cria ASs para o IPSec.

O IKE é um protocolo complexo baseado em três outros protocolos: Oakley, SKEME e ISAKMP, conforme mostra a Figura 10.58.

Troca de Chaves da Internet (IKE)

Protocolo de Gerenciamento de Chaves e Associações de Segurança da Internet (ISAKMP)

Oakley

SKEME

Figura 10.58 Componentes do IKE.

O protocolo **Oakley** foi desenvolvido por Hilarie Orman. Trata-se de um protocolo de criação de chaves. O protocolo **SKEME***, projetado por Hugo Krawcyzk, é outro protocolo para troca de chaves. Ele usa criptografia de chave pública para autenticar entidades em um protocolo de troca de chaves.

O **Protocolo de Gerenciamento de Chaves e Associações de Segurança da Internet** (ISAKMP – Internet Security Association and Key Management Protocol) é um protocolo desenvolvido pela **Agência de Segurança Nacional** (NSA – National Security Agency) que de fato implementa as trocas definidas no IKE. O ISAKMP define vários pacotes, protocolos e parâmetros que permitem que as transferências feitas no IKE sejam realizadas usando mensagens padronizadas, formatadas para criar ASs. Deixamos a discussão desses três protocolos para livros dedicados ao tema de segurança.

Rede Privada Virtual

Uma das aplicações do IPSec é a criação de redes privadas virtuais. Uma **Rede Virtual Privada** (VPN – Virtual Private Network) é uma tecnologia que vem ganhando popularidade em grandes organizações que usam a Internet global para a comunicação intra e interorganizacional, mas que exigem privacidade na comunicação interorganizacional. Uma VPN é uma rede privada, porém virtual. Ela é privada, pois garante privacidade dentro da organização. É virtual porque não usa WANs privadas reais; a rede é fisicamente pública, porém virtualmente privada. A Figura 10.59 ilustra a ideia de uma rede privada virtual. Os roteadores R1 e R2 usam a tecnologia de VPN para garantir privacidade à organização. A tecnologia de VPN usa o protocolo ESP do IPSec no modo túnel. Um datagrama privado, incluindo o cabeçalho, é encapsulado em um pacote ESP. O roteador na borda da rede do emissor usa seu próprio endereço IP e o endereço do roteador na rede de destino no novo datagrama. A rede pública (Internet) é responsável pelo transporte do pacote de R1 até R2. Entidades externas não conseguem decifrar o conteúdo do pacote ou os endereços de origem e de destino. A decifração acontece em R2, que determina o endereço de destino do pacote e o entrega.

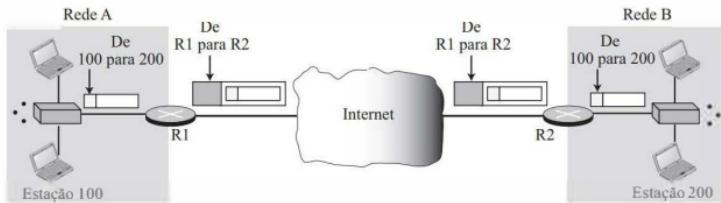



Figura 10.59 Rede privada virtual (VPN).

10.5 *FIREWALLS*

Nenhuma das medidas de segurança anteriores é capaz de impedir que Eva envie uma mensagem prejudicial ao sistema. Para controlar o acesso a um sistema, precisamos de *firewalls*. Um ***firewall*****

* N. de T.: O termo SKEME pode ser visto como uma sigla para *Secure Key Exchange Mechanism* (Mecanismo Seguro de Troca de Chaves), ou então como uma corruptela da palavra *scheme* ("esquema", em inglês).

** N. de T.: O termo *firewall* é, por vezes, traduzido para o português como "parede de fogo" ou "parede corta-fogo". Porém, preferimos manter a palavra em inglês porque ela é geralmente mais utilizada por profissionais da área de segurança.

é um dispositivo (geralmente um *software* residente em um roteador ou computador) instalado entre a rede interna de uma organização e o restante da Internet. Ela é projetada para encaminhar alguns pacotes e filtrar (não encaminhar) outros. A Figura 10.60 mostra um *firewall*.

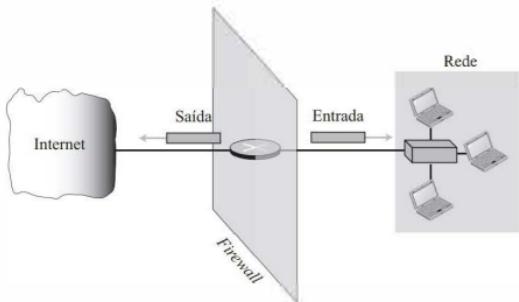



Figura 10.60 *Firewall*

Por exemplo, um *firewall* pode filtrar, na entrada, todos os pacotes destinados a um *host* ou a um servidor específico, tal como um servidor HTTP. Um *firewall* pode ser usado para prevenir o acesso a um *host* ou a um serviço específico dentro da organização. Um *firewall* é normalmente classificado como um *firewall de filtragem de pacotes* ou como um *firewall proxy*.

10.5.1 *Firewall* de filtragem de pacotes

Um *firewall* pode ser usado como um filtro de pacotes. Ele pode encaminhar ou bloquear pacotes com base nas informações presentes nos cabeçalhos da camada de rede e da camada de transporte: endereços IP de origem e destino, endereços de porta de origem e de destino e tipo de protocolo (TCP ou UDP). Um ***firewall* de filtragem de pacotes** é um roteador que usa uma tabela de filtragem para decidir quais pacotes devem ser descartados (não encaminhados). A Figura 10.61 mostra um exemplo de uma tabela de filtragem para esse tipo de *firewall*.

Interface	IP de origem	Porta de origem	IP de destino	Porta de destino
1	131.34.0.0	*	*	*
1	*	*	*	23
1	*	*	194.78.20.8	*
2	*	*	*	80

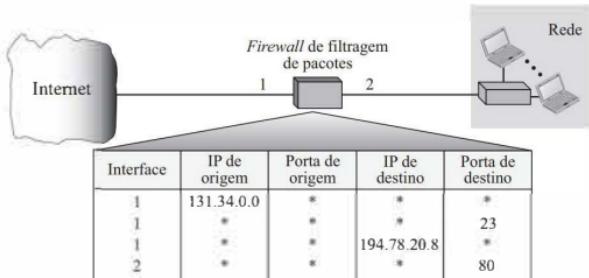



Figura 10.61 *Firewall* de filtragem de pacotes.

De acordo com a figura, os seguintes pacotes são filtrados:

1. Pacotes recebidos provenientes da rede 131.34.0.0 são bloqueados (uma precaução de segurança). Observe que o * (asterisco) significa "qualquer".
2. Os pacotes recebidos destinados a qualquer servidor TELNET interno (porta 23) são bloqueados.
3. Pacotes recebidos destinados ao *host* interno 194.78.20.8 são bloqueados. A organização quer que esse *host* seja de uso interno apenas.
4. Os pacotes de saída destinados a um servidor HTTP (porta 80) são bloqueados. A organização não quer que os funcionários naveguem na Internet.

Um *firewall* de filtragem de pacotes filtra na camada de rede ou de transporte.

10.5.2 *Firewall Proxy*

O *firewall* de filtragem de pacotes se baseia nas informações disponíveis na camada de rede e nos cabeçalhos da camada de transporte (IP e TCP/UDP). Entretanto, em algumas ocasiões, precisamos filtrar uma mensagem com base nas informações disponíveis na própria mensagem (na camada de aplicação). Por exemplo, suponha que uma organização deseja instaurar as seguintes políticas com relação às suas páginas Web: apenas os usuários da Internet que tenham estabelecido relações comerciais com a empresa anteriormente podem ter acesso; o acesso por outros usuários deve ser bloqueado. Nesse caso, o uso de um *firewall* de filtragem de pacotes não é viável porque ele não é capaz de distinguir entre diferentes pacotes chegando à porta 80 do TCP (HTTP). O teste deve ser feito na camada de aplicação (usando URLs).

Uma solução é instalar um computador *proxy* (também chamado ***gateway* de aplicação**), que é colocado entre o computador do cliente e o computador da corporação. Quando o processo-cliente do usuário enviar uma mensagem, o *gateway* de aplicação executa um processo-servidor para receber o pedido. O servidor abre o pacote no nível da aplicação e verifica se o pedido é legítimo. Se for, o servidor atua como um processo-cliente e envia a mensagem para o servidor de fato dentro da corporação. Caso contrário, a mensagem é descartada e uma mensagem de erro é enviada ao usuário externo. Desse modo, os pedidos dos usuários externos são filtrados com base no conteúdo dos pacotes na camada de aplicação. A Figura 10.62 mostra uma implementação de um *gateway* de aplicação para mensagens HTTP.

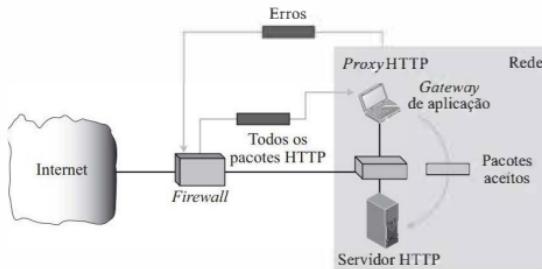



Figura 10.62 *Firewall proxy*

Um *firewall proxy* filtra na camada de aplicação.

10.6 MATERIAL DO FINAL DO CAPÍTULO

10.6.1 Leitura adicional

Diversos livros dão uma cobertura bastante completa sobre criptografia e segurança de redes: [For 08], [Sta 06], [Bis 05], [Mao 04], [Sti 06]. [Res 01], [Tho 00], [DH 03] e [Gar 95].

10.6.2 Termos-chave

- algoritmo de cifração
- algoritmo de decifração
- Algoritmo de *Hash* Seguro (SHA – Secure Hash Algorithm)
- assinatura digital
- Associações de Segurança (AS)
- autenticação por desafio-resposta
- Autoridade Certificadora (AC)
- Base de Dados de Associações de Segurança (SAD – Security Association Database)
- Base de Dados de Políticas de Segurança (SPD – Security Policy Database)
- bilhete (*ticket*)
- caixa-P (*P-box*)
- caixa-S (*S-box*)
- Centro de Distribuição de Chaves (KDC – Key-Distribution Center)
- certificado de chave pública
- chave privada
- chave pública
- cifra
- cifra aditiva
- cifra de autochave
- cifra de bloco
- cifra de César
- cifra de chave assimétrica
- cifra de chave simétrica
- cifra de deslocamento
- cifra de fluxo
- cifra de substituição
- cifra de transposição
- cifra de uso único (*one-time pad*)
- cifra monoalfabética
- cifra polialfabética
- cifração
- Código de Autenticação de Mensagens (MAC – Message Authentication Code)
- conjunto de algoritmos criptográficos (*cipher suite*)
- criptografia
- decifração
- Encapsulamento de Dados de Segurança (ESP – Encapsulating Security Payload)
- esteganografia
- Extensões Seguras/Multifunção para Mensagens de Internet (S/MIME – Secure/Multipurpose Internet Mail Extension)
- *firewall*
- *firewall* de filtragem de pacotes
- *firewall proxy*
- função de *hash* criptográfico
- *gateway* de aplicação
- MAC baseado em *hash* (HMAC – *hashed MAC*)
- modo de transporte
- modo túnel
- Negação de Serviço (DoS – Denial of Service)
- Oakley
- Padrão de Assinatura Digital (DSS – Digital Signature Standard)
- Padrão de Cifração de Dados (DES – Data Encryption Standard)
- Política de Segurança
- Privacidade Bastante Boa (PGP – Pretty Good Privacy)
- Protocolo de Apresentação (*Handshake*)
- Protocolo de Cabeçalho de Autenticação (AH – Authentication Header)
- Protocolo de Gerenciamento de Chaves e Associações de Segurança da Internet (ISAKMP – Internet Security Association and Key Management Protocol)

- protocolo Diffie-Hellman
- protocolo SSL (*Secure Sockets Layer*, ou Camada de Sockets Segura)
- protocolo TLS (*Transport Layer Security* ou Segurança da Camada de Transporte)
- rede de confiança (*web of trust*)
- Rede Virtual Privada (VPN – Virtual Private Network)
- resumo criptográfico
- Resumo Criptográfico da Mensagem (MD – Message Digest)
- Segurança IP (IPSec – IP Security)
- Sintaxe para Mensagens Cifradas (CMS – Cryptographic Message Syntax)
- sistema criptográfico RSA
- SKEME
- texto às claras
- texto cifrado
- Troca de Chaves da Internet (IKE – Internet Key Exchange)
- X.509

10.6.3 Resumo

As três metas de segurança podem ser ameaçadas por ataques contra a segurança. Duas técnicas foram desenvolvidas para proteger informações contra ataques: a criptografia e a esteganografia.

Em uma cifra de chave simétrica, a mesma chave é utilizada para cifrar e para decifrar, e a chave pode ser usada para comunicações bidirecionais. Podemos dividir as cifras de chave simétrica tradicionais em duas grandes categorias: cifras de substituição e cifras de transposição.

Na criptografia de chave assimétrica, existem duas chaves distintas: uma pública e uma privada. O termo criptografia de chave assimétrica indica que Bob e Alice não podem usar o mesmo conjunto de chaves para comunicações bidirecionais.

Outros aspectos de segurança incluem integridade, autenticação de mensagens, autenticação de entidades e gerenciamento de chaves.

O protocolo Privacidade Bastante Boa (PGP – Pretty Good Privacy), inventado por Phil Zimmermann, fornece privacidade, autenticação e integridade a mensagens de e-mail. Outro serviço de segurança projetado para uso no correio eletrônico é o Extensões Seguras/Multifunção para Mensagens de Internet (S/MIME – Secure/Multipurpose Internet Mail Extension).

Um protocolo de segurança da camada de transporte fornece serviços de segurança fim a fim para aplicações que usam os serviços de um protocolo confiável da camada de transporte, como o TCP. Dois protocolos são dominantes atualmente para fornecer segurança na camada de transporte: a Camada de Sockets Segura (SSL – Secure Sockets Layer) e o Segurança da Camada de Transporte (TLS – Transport Layer Security).

O Segurança IP (IPSec – IP Security) consiste em uma coleção de protocolos projetados pela IETF para fornecer segurança a pacotes no nível da rede. O IPSec opera no modo de transporte ou no modo túnel. O IPSec define dois protocolos: o protocolo AH (*Authentication Header*, ou Cabeçalho de Autenticação) e o protocolo ESP (*Encapsulating Security Payload*, ou Encapsulamento de Dados de Segurança).

Um *firewall* é um dispositivo (geralmente um roteador ou um computador) instalado entre a rede interna de uma organização e o restante da Internet. Ele é projetado para encaminhar alguns pacotes e filtrar outros. Um *firewall* é normalmente classificado como um *firewall* de filtragem de pacotes ou um *firewall proxy*.

10.7 ATIVIDADES PRÁTICAS

10.7.1 Testes

Diversos testes interativos relativos a este capítulo podem ser encontrados no *site* www.mhhe.com/forouzan. Recomendamos que o estudante realize os testes para verificar o seu entendimento dos materiais antes de continuar com as atividades práticas.

Questões

10-1 Qual dos seguintes ataques é uma ameaça à confidencialidade?

a. escuta **b.** falsificação **c.** retratação

10-2 Qual dos seguintes ataques é uma ameaça à integridade?

a. modificação

b. repetição

c. negação de serviço

10-3 Qual dos seguintes ataques é uma ameaça à disponibilidade?

a. retratação

b. negação de serviço

c. modificação

10-4 Qual das seguintes palavras significa "escrita secreta"? Qual significa "escrita oculta"?

a. criptografia **b.** esteganografia

10-5 Quando uma carta selada é enviada de Alice para Bob, isto é um exemplo de uso de criptografia ou de esteganografia para fornecer confidencialidade à mensagem?

10-6 Quando uma carta é enviada de Bob para Alice em um idioma que só os dois podem entender, isto é um exemplo de criptografia ou de esteganografia?

10-7 Alice encontrou uma maneira de escrever para Bob secretamente. Cada vez que ela escreve, ela utiliza um novo texto, tal como um artigo de jornal, mas insere um ou dois espaços entre as palavras. Um espaço simples significa um dígito binário 0; um espaço duplo significa um dígito binário 1. Bob extrai os dígitos binários e os interpreta usando o código ASCII. Esse é um exemplo de criptografia ou de esteganografia? Explique.

10-8 Alice e Bob trocam mensagens confidenciais. Eles compartilham um número muito grande como a chave de cifração e decifração em ambas as direções. Esse é um exemplo de criptografia de chave simétrica ou de chave assimétrica? Explique.

10-9 Alice usa a mesma chave quando ela cifra uma mensagem a ser enviada para Bob e quando decifra uma mensagem recebida de Bob. Esse é um exemplo de criptografia de chave simétrica ou de chave assimétrica? Explique.

10-10 Quais são as diferenças e semelhanças entre uma cifra de substituição e uma cifra de transposição?

10-11 Em uma cifra, todos os As do texto às claras foram transformados em Ds no texto cifrado e todos os Ds no texto às claras foram transformados em Hs no texto cifrado. Essa é uma cifra de substituição monoalfabética ou polialfabética? Explique.

10-12 Qual cifra pode ser quebrada mais facilmente, a monoalfabética ou a polialfabética?

10-13 Considere que Alice e Bob usem uma cifra aditiva baseada em aritmética módulo 26. Se Eva, a atacante, deseja quebrar a cifra tentando todas as chaves possíveis (ataque de força bruta), quantas chaves ela precisa testar, em média?

10-14 Se tivermos um único número inteiro como chave nos Exemplos 10.1 e 10.2 do texto, quantas chaves temos no Exemplo 10.3 do texto?

10-15 Considere que tenhamos um texto às claras de 1.000 caracteres. Quantas chaves precisamos para cifrar ou decifrar a mensagem em cada uma das seguintes cifras?

a. aditiva

b. monoalfabética

c. autochave

10-16 De acordo com as definições de cifras de fluxo e de bloco, diga qual das cifras a seguir é uma cifra de fluxo.

a. aditiva

b. monoalfabética

c. autochave

10-17 Um bloco de permutação (caixa-P) em uma cifra de bloco moderna apresenta cinco entradas e cinco saídas. Esta é uma permutação _____?

a. simples

b. com compressão

c. com expansão

10-18 Um bloco de permutação (caixa-P) em uma cifra de bloco moderna é um exemplo de uma cifra de transposição sem chave. O que essa afirmação significa? (Ver Figura 10.8 no texto.)

10-19 Em uma cifra de bloco moderna, geralmente precisamos usar um componente no algoritmo de decifração que é o inverso do componente

usado no algoritmo de cifração. Qual é o inverso de cada um dos seguintes componentes?

a. permuta

b. deslocamento para a direita

c. combinação

10-20 Em cada rodada do DES, temos todos os componentes definidos na Figura 10.8 do texto. Quais componentes usam uma chave e quais componentes não o fazem?

10-21 Na Figura 10.10 no texto, por que precisamos de uma caixa-P com expansão? Por que não podemos usar uma caixa-P simples ou com compressão?

10-22 A Figura 10.9 do texto mostra que o DES cria 16 chaves distintas de 48 *bits*, uma para cada rodada. Por que precisamos de 16 chaves distintas? Por que não podemos usar a mesma chave em cada rodada?

10-23 Se a cifra *one-time pad* (Figura 10.12 do texto) é a cifra mais simples e segura, por que ela não é usada o tempo todo?

10-24 Se Alice e Bob precisam se comunicar usando criptografia de chave assimétrica, de quantas chaves eles precisam? Quem precisa criar essas chaves?

10-25 Por que a criptografia de chave assimétrica é utilizada apenas com mensagens pequenas?

10-26 Em uma cifra de chave assimétrica, qual chave é usada para a cifração? Qual chave é usada para a decifração?

a. chave pública **b.** chave privada

10-27 Na RSA, por que Bob não pode escolher o valor 1 como sua chave pública *e*?

10-28 Qual é o papel da chave secreta adicionada à função de *hash* na Figura 10.17 do texto (MAC)? Explique.

10-29 Quais são as diferenças e semelhanças entre a autenticação de mensagem e a autenticação de entidades?

10-30 Alice assina a mensagem que ela envia a Bob para provar que é a remetente da mensagem. Qual das seguintes chaves Alice deve usar nesse caso?

a. A chave pública de Alice

b. A chave privada de Alice

10-31 Alice precisa enviar uma mensagem para um grupo de cinquenta pessoas. Se Alice precisa usar o serviço de autenticação de mensagem, qual dos seguintes esquemas você recomendaria?

a. MAC **b.** assinatura digital

10-32 Qual dos seguintes serviços não é fornecido por esquemas de assinatura digital?

a. autenticação de mensagens

b. confidencialidade

c. irretratabilidade

10-33 Considere que Alice precisa enviar um documento confidencial assinado para 100 pessoas. Quantas chaves Alice precisa usar para criar 100 cópias desse documento se ela usa um esquema de chave assimétrica para obter o serviço de confidencialidade? Explique.

10-34 Em um clube com 50 membros, quantas chaves secretas são necessárias para permitir a troca de mensagens secretas entre qualquer par de membros?

10-35 O conceito de Centro de Distribuição de Chaves (KDC) foi desenvolvido para resolver o problema da distribuição de chaves ________.

a. secretas **b.** públicas **c.** privadas

10-36 O conceito de Autoridade Certificadora (AC) foi desenvolvido para resolver o problema da distribuição de chaves ________.

a. secretas

b. públicas

c. privadas

Problemas

10-1 Determine o tipo de ataque em cada um dos seguintes casos:

a. Um estudante invade o escritório de um professor para obter uma cópia do próximo teste.

b. Um estudante emite um cheque de R$ 10 para comprar um livro seminovo. Mais tarde, o estudante descobre que o cheque foi descontado por R$ 100.

c. Um estudante envia centenas de e-mails por dia para a escola usando um endereço de e-mail falso.

10-2 Use a cifra aditiva com k = 10 para cifrar o texto às claras "livro". Em seguida, decifre

a mensagem para recuperar o texto às claras original.

10-3 Cifre a mensagem "eis um exemplo", usando uma cifra aditiva com chave = 20. Ignore o espaço entre as palavras. Decifre a mensagem para recuperar o texto às claras original.

10-4 Atbash era uma cifra popular entre os escritores bíblicos. No Atbash, "A" é cifrado como "Z", "B" é cifrado como "Y" e assim por diante. Da mesma forma, "Z" é cifrado como "A", "Y" é cifrado como "B", e assim por diante. Considere que o alfabeto seja dividido na metade e que as letras na primeira metade sejam cifradas como as letras na segunda metade e vice-versa. Determine o tipo de cifra e a chave. Cifre o texto às claras "um exemplo" usando a cifra Atbash.

10-5 Uma cifra de substituição não precisa consistir em uma transformação de caracteres para caracteres. Em uma cifra de Políbio, cada letra do texto às claras é cifrada como dois inteiros. A chave é uma matriz de caracteres de dimensões 5 × 5. O texto às claras é o caractere na matriz, enquanto o texto cifrado corresponde aos dois inteiros (cada um deles entre 1 e 5) representando os números de linha e de coluna. Cifre a mensagem "Um exemplo" usando a cifra de Políbio com a seguinte chave:

	1	2	3	4	5
1	z	q	p	f	e
2	y	r	o	g	d
3	x	s	n	h	c
4	w	t	m	i/j	b
5	v	u	l	k	a

10-6 Alice pode usar apenas uma cifra aditiva em seu computador para enviar uma mensagem a um amigo. Ela acredita que a mensagem estará mais segura se ela cifrar tal mensagem duas vezes, cada vez com uma chave diferente. Ela está correta? Justifique sua resposta.

10-7 Um dos ataques que um atacante pode empregar contra uma cifra simples como uma cifra aditiva é o chamado ataque de *texto cifrado apenas*. Nesse tipo de ataque, o atacante intercepta o texto cifrado e tenta determinar a chave e, por fim, o texto às claras correspondente. Um dos métodos usados em um ataque de texto cifrado puro é o chamado ataque de *força bruta*, no qual o atacante testa várias chaves e decifra a mensagem até que tal mensagem venha a fazer sentido. Considere que o atacante tenha interceptado a mensagem cifrada "UVACLYZLJBYL". Tente decifrar a mensagem utilizando chaves iniciando em 1 até que apareça um texto às claras que faça sentido (em inglês).

10-8 Outro método usado em um ataque de texto cifrado apenas (ver problema anterior) é denominado abordagem *estatística*, na qual o atacante intercepta uma longa mensagem cifrada e tenta analisar as estatísticas dos caracteres no texto cifrado. Uma cifra simples como a cifra aditiva não altera as estatísticas dos caracteres, pois a cifração é um para um. Considere que o atacante tenha interceptado a seguinte mensagem cifrada e que o caractere "e" seja o caractere mais comum em um texto às claras em inglês (a língua na qual o texto às claras foi escrito). Use essa informação para determinar a chave da cifra e decifrar o texto cifrado.

```
XLILSYWIMWRSAJSVWEPIJSVJSYVQ
MPPMSRHSPPEVWMXMWASVXLQSVIL
YVVCFIJSVIXLIWIPPIVVIGIMZIW
QSVISJJIVW
```

10-9 Em uma cifra de transposição, as chaves de cifração e de decifração são muitas vezes representadas como duas tabelas unidimensionais (vetores) e a cifra é representada como um programa de *software*.

a. Determine o vetor correspondente à chave de cifração da Figura 10.6, apresentada no texto. Dica: o valor de cada elemento pode revelar o número da coluna de entrada; já o índice pode revelar o número da coluna de saída.

b. Determine o vetor correspondente à chave de decifração da Figura 10.6, apresentada no texto.

c. Explique como, dada a chave de cifração, é possível determinar a chave de decifração.

10-10 A operação de deslocamento circular é um dos componentes das cifras de bloco modernas.

a. Mostre o resultado de um deslocamento circular de 3 *bits* para a esquerda aplicado sobre a palavra $(10011011)_2$.

b. Mostre o resultado de um deslocamento circular de 3 *bits* para a direita aplicado sobre o resultado do item *a*.

c. Compare o resultado do item *b* com a palavra original dada no item *a* para mostrar que as operações de deslocamento para a direita e para a esquerda são inversas entre si.

10-11 A operação de permuta é um dos componentes das cifras de bloco modernas.

a. Aplique a operação de permuta sobre a palavra $(10011011)_2$.

b. Aplique a operação de permuta sobre a palavra resultante do item *a*.

c. Compare os resultados dos itens *a* e *b* para mostrar que a operação de permuta é autoinversível.

10-12 Uma operação muito comum em cifras de bloco é a operação de XOR (OU-exclusivo). Determine o resultado das seguintes operações. Interprete os resultados.

a. $(01001101) \oplus (01001101)$

b. $(01001101) \oplus (00000000)$

10-13 Suponha que você queira escrever um programa para simular as caixas de permutação da Figura 10.8, apresentada no texto.

a. Mostre como você pode representar cada caixa na forma de uma tabela.

b. Mostre a inversão de cada caixa na forma de uma tabela.

10-14 Considere que tenhamos uma caixa de substituição (caixa-S, ou *S-box*) sem chave com três entradas (x_1, x_2 e x_3) e duas saídas (y_1 e y_2). A relação entre as entradas e as saídas é definida da seguinte forma ($\oplus$ significa XOR, ou OU-exclusivo):

$$y_1 = x_1 \oplus x_2 \oplus x_3 \qquad y_2 = x_1$$

Qual é a saída se a entrada for (110)?
Qual é a saída se a entrada for (001)?

10-15 Cada rodada em uma cifra de bloco deve ser inversível para que a cifra como um todo seja inversível. Cifras de bloco modernas usam duas abordagens para atingir esse objetivo. Na primeira abordagem, cada componente é inversível; na segunda abordagem, alguns componentes não são inversíveis, mas a rodada como um todo é inversível usando algo conhecido como estrutura de Feistel. Essa abordagem é usada no DES, descrito no texto. O truque da cifra Feistel é usar a operação de XOR como um dos componentes. Para enxergar esse ponto, suponha que uma rodada inclua um componente não inversível, NI, e uma operação de XOR, conforme mostra a Figura 10.63. Prove que a rodada inteira é inversível, o que significa que o texto às claras pode ser recuperado a partir do texto cifrado. Dica: use as propriedades da operação de XOR ($x \oplus x = 0$ e $x \oplus 0 = x$).

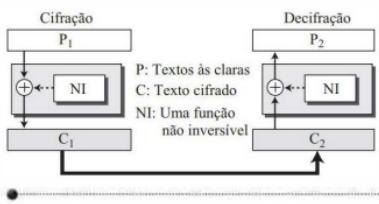



Figura 10.63 Esquema para o Problema 10-15.

10-16 Na Figura 10.9, temos um permutador em cada rodada. Qual é a utilidade desse permutador?

10-17 Na Figura 10.9, temos duas operações de permutação simples: uma *permutação inicial* e uma *permutação final*. Os especialistas acreditam que essas operações são inúteis e não ajudam a tornar a cifra mais resistente a ataques. Você é capaz de determinar a razão para essa afirmação?

10-18 A chave do DES apresenta 56 *bits*. Considere que Eva, a atacante, tente determinar a chave usando um ataque de força bruta (testando todas as chaves, uma a uma). Se ela é capaz de testar um milhão de chaves (aproximadamente 2^{20}) por segundo (usando um computador poderoso), quanto tempo ela levará para quebrar a cifra?

10-19 Considerando que Bob, usando o sistema criptográfico RSA, selecione $p = 11$, $q = 13$ e $d = 7$, qual dos seguintes itens pode corresponder ao valor de sua chave pública e?

a. 11 **b.** 103 **c.** 19

10-20 No RSA, dados $p = 107$, $q = 113$, $e = 13$ e $d = 3653$, cifre a mensagem "ISTO É DIFÍCIL", usando o intervalo 00-26 (A: 00, espaço: 26 e ignore os acentos) como o esquema de codificação. Decifre o texto cifrado para recuperar a mensagem original.

10-21 Uma função de ***hash*** criptográfico deve ser *resistente à segunda inversão*, o que significa que, dada a mensagem M e o resumo

criptográfico da mensagem *d*, não devemos ser capazes de encontrar qualquer outra mensagem, M', cujo resumo criptográfico seja *d*. Em outras palavras, duas mensagens diferentes não podem apresentar o mesmo resumo criptográfico. Considerando esse requisito, mostre que uma soma de verificação (*checksum*) tradicional usada na Internet não pode ser usada como uma função de *hash*.

10-22 Explique por que chaves públicas e privadas não podem ser utilizadas na geração de um MAC.

10-23 O *nonce* na Figura 10.22 é usado para evitar um possível ataque de repetição da terceira mensagem. Eva não consegue reapresentar a terceira mensagem fingindo que esse é um novo pedido de Alice, pois quando Bob receber a resposta, o valor de R_B não será mais válido. Isto significa que podemos eliminar a primeira e a segunda mensagens se adicionarmos um carimbo de tempo (*timestamp*) ao diagrama. Mostre uma nova versão da Figura 10.22 usando um carimbo de tempo.

10-24 Explique por que a cifração é usada na segunda mensagem (de Bob para Alice) na Figura 10.23, porém uma assinatura é criada na terceira mensagem (de Alice para Bob) na Figura 10.24.

10-25 A Figura 10.22 mostra uma autenticação unidirecional que autentica Alice em relação a Bob. Modifique essa figura para fornecer autenticação bidirecional: para autenticar Alice em relação a Bob e Bob em relação a Alice.

10-26 Modifique a Figura 10.23 para permitir uma autenticação bidirecional. Alice precisa ser autenticada em relação a Bob e Bob em relação a Alice.

10-27 Modifique a Figura 10.24 para permitir uma autenticação bidirecional. Alice precisa ser autenticada em relação a Bob e Bob em relação a Alice.

10-28 Você talvez tenha notado que há uma falha na Figura 10.26. Eva, a atacante, pode reenviar a terceira mensagem e, se ela de alguma forma puder ganhar acesso à chave de sessão, pode fingir ser Alice e trocar mensagens com Bob. O problema pode ser evitado se Alice e Bob usarem dois *nonces*. Lembre-se que *nonces* têm um tempo de vida e sua finalidade principal é evitar ataques de repetição. Modifique a Figura 10.26 de modo a adicionar dois *nonces*.

10-29 Considere que temos um resumo criptográfico muito simples. Nosso pouco realista resumo criptográfico de mensagens consiste simplesmente em um número entre 0 e 25. O resumo criptográfico é inicialmente fixado em 0. A função de *hash* criptográfico soma o valor atual do resumo criptográfico ao valor do caractere que está sendo lido (entre 0 e 25). A soma é feita usando aritmética módulo 26. Qual é o valor do resumo criptográfico se a mensagem for "HELLO"? Por que esse resumo criptográfico não é seguro?

10-30 Para entender o conceito de distribuição de chaves secretas, considere que um pequeno clube privado tenha apenas 100 membros (excluindo o presidente). Responda às seguintes perguntas:

a. Quantas chaves secretas são necessárias caso todos os membros do clube precisem enviar mensagens secretas uns aos outros?

b. Quantas chaves secretas são necessárias se todos os membros confiam no presidente do clube? Nesse caso, se um membro tiver que enviar uma mensagem para outro membro, ele primeiro envia a mensagem para o presidente; o presidente, então, envia a mensagem para outro membro.

c. Quantas chaves secretas são necessárias se o presidente decidir que dois membros que precisam se comunicar devem primeiro entrar em contato com ele? O presidente, então, cria uma chave temporária para ser usada entre os dois. A chave temporária é cifrada e enviada a ambos os membros.

10-31 Apresentamos dois serviços de segurança para mensagens de e-mail (PGP e S/MIME). Explique por que aplicações de correio eletrônico não podem usar os serviços do SSL/TLS, precisando usar PGP ou S/MIME.

10-32 Considere que Alice queira enviar um e-mail para Bob. Explique como a integridade do e-mail é obtida usando o PGP.

10-33 Considere que Alice queira enviar um e-mail para Bob. Explique como a confidencialidade do e-mail é obtida usando o PGP.

10-34 Considere que Alice queira enviar um e-mail para Bob. Explique como a integridade do e-mail é obtida usando o S/MIME.

10-35 Considere que Alice queira enviar um e-mail para Bob. Explique como a autenticação do e-mail é obtida usando o S/MIME.

10-36 Considere que Alice queira enviar um e-mail para Bob. Explique como a confidencialidade do e-mail é obtida usando o S/MIME.

10-37 Quando falamos sobre autenticação no SSL, queremos dizer *autenticação de mensagens* ou *autenticação de entidades*? Explique.

10-38 Quando falamos sobre autenticação no PGP (ou S/MIME), queremos dizer *autenticação de mensagens* ou *autenticação de entidades*? Explique.

10-39 Se os algoritmos de criptografia no PGP e no S/MIME não podem ser negociados, como o receptor do e-mail determina qual algoritmo foi usado pelo remetente?

10-40 Podemos usar SSL com UDP? Explique.

10-41 Por que não existe a necessidade de uma Associação de Segurança no SSL?

10-42 Compare e diferencie o PGP e o S/MIME. Quais são as vantagens e desvantagens de cada um?

10-43 A execução do Protocolo de Apresentação no SSL deve ocorrer antes ou depois do protocolo *three-way handshake* do TCP? Eles podem ser combinados? Explique.

10-44 As estações A e B usam IPSec no modo de transporte. Podemos dizer que as duas estações precisam criar um serviço virtual orientado à conexão entre elas? Explique.

10-45 Quando falamos sobre autenticação no IPSec, queremos dizer *autenticação de mensagens* ou *autenticação de entidades*? Explique.

10-46 Se Alice e Bob estão enviando mensagens continuamente um ao outro, eles podem criar uma associação de segurança uma só vez e usá-la para todos os pacotes trocados? Explique.

10.8 EXPERIMENTOS DE SIMULAÇÃO

10.8.1 *Applets*

Criamos alguns *applets* Java para demonstrar alguns dos principais conceitos discutidos neste capítulo. Recomendamos que o estudante ative esses *applets* no *site* www.mhhe.com/forouzan e examine cuidadosamente os protocolos em ação.

10.8.2 Experimentos de laboratório

Nesta seção, usamos o Wireshark para simular dois protocolos: o protocolo Ethernet e o protocolo ARP. As descrições completas desses experimentos de laboratório encontram-se no *site* www.grupoa.com.br.

Lab10-1 Estudamos FTP e TELNET no Capítulo 2. A transferência de arquivos usando FTP e o acesso remoto a sistemas usando TELNET não são seguros. No Capítulo 2, também aprendemos que podemos usar o Secure Shell (SSH) para simular tanto o FTP como o TELNET. Neste laboratório, queremos usar o SSH e capturar os pacotes usando o Wireshark para entender como os protocolos de segurança da Internet, como o SSL/TLS, sobre os quais aprendemos neste capítulo, podem permitir a transferência de arquivos e o acesso remoto a sistemas de forma segura.

Lab10-2 No Capítulo 2, aprendemos sobre o HTTP, o protocolo usado para acessar páginas da Internet. O HTTP, por si só, não oferece mecanismos de segurança. No entanto, podemos combinar o HTTP e o SSL/TLS para adicionar segurança ao HTTP. O novo protocolo é denominado Protocolo de Transferência de Hipertexto Seguro (HTTPS – HyperText Transfer Protocol Secure). Neste laboratório, queremos usar o HTTPS e capturar os pacotes usando o Wireshark para examinar o conteúdo dos pacotes SSL/TLS quando utilizados com o HTTPS.

Lab10-3 No Capítulo 4, aprendemos sobre o protocolo IP. Neste laboratório, queremos usar o IPsec para criar uma conexão IP segura entre duas estações finais.

10.9 TAREFAS DE PROGRAMAÇÃO

Escreva o código-fonte, compile e teste os seguintes programas utilizando a linguagem de programação de sua preferência:

Prg10-1 Um programa geral que implementa uma cifra aditiva (cifração e decifração). A entrada para o programa é um indicador que define se a operação será de cifração ou decifração, a chave simétrica e o texto às claras ou o texto cifrado. A saída é o texto cifrado ou às claras, dependendo do tipo de operação requisitado.

Prg10-2 Um programa genérico que implemente uma cifra de transposição (cifração e decifração). A entrada para o programa é um indicador que define se a operação será de cifração ou decifração, a chave simétrica e o texto às claras ou o texto cifrado. A saída é o texto cifrado ou às claras, dependendo do tipo de operação requisitado.

Prg10-3 Um programa geral que implementa o sistema criptográfico RSA. A entrada para o programa é um indicador que define se a operação será de cifração ou decifração, os valores de p e q, o valor de e e o texto às claras ou o texto cifrado. A saída é o texto cifrado ou às claras, dependendo do tipo de operação requisitado.

CAPÍTULO

11 PROGRAMAÇÃO DE *SOCKETS* EM JAVA

No Capítulo 2, discutimos a criação de programas cliente-servidor usando a linguagem C. Neste capítulo, fazemos o mesmo usando a linguagem Java para mostrar como as entidades definidas na linguagem C são redefinidas em uma linguagem de programação orientada a objetos. Escolhemos a linguagem Java porque muitos aspectos da programação podem ser facilmente demonstrados usando as poderosas classes disponíveis em Java. Abordamos as principais questões relacionadas à programação usando a tradicional *API de interface socket*, mas a discussão pode ser estendida para outras áreas de programação de redes sem dificuldades. Consideramos que o leitor esteja familiarizado com os conceitos básicos de programação Java. O capítulo é dividido em três seções.

- Na primeira seção, mostramos como entidades, como endereços IP, portas e endereços de *socket* são representadas por classes correspondentes em Java. Fornecemos alguns programas para mostrar como podemos usar essas entidades. Também revisamos o conceito do paradigma cliente-servidor que discutimos no Capítulo 2.
- Na segunda seção, primeiramente apresentamos as classes Java que são usadas na programação usando UDP. Mostramos, então, como podemos escrever programas cliente--servidor simples usando a abordagem iterativa. Em seguida, mostramos como podemos modificar o programa-servidor utilizando a abordagem concorrente. Também explicamos como os programas genéricos escritos nesta seção podem ser usados para fornecer serviços por meio de alguns exemplos.
- Na terceira seção, apresentamos as classes Java que são usadas na programação usando TCP. Em seguida, mostramos como podemos escrever programas cliente-servidor simples usando a abordagem iterativa. Finalmente, explicamos como é possível modificar o programa-servidor utilizando a abordagem concorrente. Também mostramos como os programas genéricos escritos nesta seção podem ser usados para fornecer serviços por meio de alguns exemplos.

11.1 INTRODUÇÃO

Nesta seção, discutimos como as ideias gerais da programação de redes usando C, sobre as quais discutimos no Capítulo 2, podem ser usadas na programação de redes usando Java.

11.1.1 Endereços e portas

Qualquer que seja a linguagem utilizada, a programação de redes definitivamente precisa lidar com endereços IP e números de porta. Apresentamos brevemente como endereços e portas são

representados em Java. Recomendamos que o leitor compare as representações dessas duas entidades em C e em Java.

Endereços IP

Conforme discutimos no Capítulo 4, existem dois tipos de endereços IP usados na Internet: endereços IPv4 (32 *bits*) e IPv6 (128 *bits*). Em Java, um endereço IP é definido como um objeto instanciado a partir da classe *InetAddress*. Essa classe foi definida originalmente como uma classe *final*, o que significa que ela não pode ser herdada (ou seja, não há classes-filhas geradas a partir dela). Mais tarde, o Java modificou essa classe e definiu duas subclasses herdeiras dela: *Inet4Address* e *Inet6Address*. No entanto, na maioria das vezes, usamos apenas a classe InetAddress para criar endereços IPv4 e IPv6. A Tabela 11.1 mostra a assinatura de alguns métodos dessa classe.

Tabela 11.1 Resumo da classe InetAddress.

```
public class java.net.InetAddress extends java.lang.Object implements Serializable

// Métodos estáticos
public static InetAddress [] getAllByName (String host) throws UnknownHostException
public static InetAddress getByName (String host) throws UnknownHostException
public static InetAddress getLocalHost () throws UnknownHostException

// Métodos de instância
public byte [] getAddress ()
public String toString ()
public String getHostAddress ()
public String getHostName ()
public String getCanonicalHostName ()
public boolean isAnyLocalAddress ()
public boolean isLinkLocalAddress ()
public boolean isLoopbackAddress ()
public boolean isMulticastAddress ()
public boolean isMCGlobal ()                          // MC significa multicast
public boolean isMCLinkLocal ()
public boolean isMCNodeLocal ()
public boolean isMCOrgLocal ()                        // Org significa organização
public boolean isMCSiteLocal ()
public boolean isReachable (int timeout)
public boolean isReachable (NetworkInterface interface, int ttl, int timeout)
```

Não há um construtor público na classe InetAddress, mas podemos usar qualquer um dos métodos estáticos dessa classe para criar uma instância de InetAddress. A classe também tem alguns métodos de instância que podem ser usados para recuperar o endereço contido no objeto em diferentes formatos ou para recuperar alguma informação sobre o objeto. Usamos apenas alguns dos métodos, mas essa referência pode ajudar a fazer alguns exercícios.

Exemplo 11.1

Neste exemplo, mostraremos como podemos usar o segundo e o terceiro métodos estáticos para obter o InetAddress de um *site* e do *host* local (Tabela 11.2).

Tabela 11.2 Esquema para o Exemplo 11.1.

```
1   import java.net.*;
2   import java.io.*;
3   public class GetIPAddress
4   {
5       public static void main (String [] args) throws IOException, UnkownHostException
6       {
7           InetAddress mysite = InetAddress.getByName ("forouzan.biz");
8           InetAddress local = InetAddress.getLocalHost ();
9           InetAddress addr = InetAddress.getByName ("23.12.71.8");
10
11          System.out.println (mysite);
12          System.out.println (local);
13          System.out.println (addr);
14
15          System.out.println (mysite.getHostAddress ());
16          System.out.println (local.getHostName ());
17      }// Fim do main
18
19  }// Fim da classe
```

Resultado:
forouzan.biz/204.200.156.162
Behrouz/64.183.101.114 / 23.12.71.8
204.200.156.162
Behrouz

Na linha 7, usamos o segundo método estático para obter o endereço IP do *site* "forouzan.biz". O programa usa o DNS para descobrir o endereço IP do *site*. Na linha 8, usamos o terceiro método estático para obter o endereço IP do *host* local no qual estamos trabalhando. Na linha 9, passamos um endereço IP, na forma de uma cadeia de caracteres (*string*), para o método getByName, de modo a transformá-lo em um objeto InetAddress.

Perceba que um objeto InetAddress não é uma cadeia de caracteres, porém ele é serializável. As linhas 11 a 13 imprimem os endereços mencionados anteriormente conforme armazenados nos objetos InetAddress: o nome do *host* seguido por uma barra seguido pelo endereço em notação decimal com pontos. No entanto, como o endereço obtido na linha 9 não tem um nome de *host* associado, a parte relativa ao *host* fica vazia.

Podemos usar o método *getHostAddress* para obter a parte correspondente ao endereço de um objeto InetAddress como um *string*, conforme mostrado na linha 15. Na linha 16, usamos o método *getHostName* para descobrir o nome de um *host* dado seu endereço (usando o DNS novamente).

Números de porta

Um número de porta na pilha de protocolos TCP/IP é um número inteiro de 16 *bits* sem sinal. No entanto, como o Java não fornece tipos de dados numéricos sem sinal, um número de porta em Java é definido como um tipo de dados inteiro (*int* de 32 *bits*) no qual os 16 *bits* à esquerda são fixados em zeros. Isto evita que um número de porta elevado seja erroneamente interpretado como um número negativo.

Exemplo 11.2

O programa na Tabela 11.3 mostra que, se usarmos uma variável do tipo *short* para armazenar um número de porta, podemos receber um valor negativo. Uma variável do tipo *int* nos fornece um valor correto.

Tabela 11.3 Esquema para o Exemplo 11.2.

```
1   import java.io.*;
2   public class Ports
3   {
4       public static void main (String [] args) throws IOException
5       {
6           short shortPort = (short) 0xFFF0;
7           System.out.println (shortPort);
8
9           int intPort = 0xFFF0;
10          System.out.println (intPort);
11      }// Fim do main
12
13  }// Fim da classe
```

Resultado:

```
–16
65520
```

InetSocketAddress

Um endereço de *socket* é uma combinação de um endereço IP e um número de porta. Em Java, existe uma classe abstrata denominada *SocketAddress*, mas a classe usada em programação de redes usando Java é a classe *InetSocketAddress* que herda da classe SocketAddress. A Tabela 11.4 mostra um resumo dos métodos dessa classe.

Tabela 11.4 Resumo da classe InetSocketAddress.

```
public class java.net. InetSocketAddress extends java.lang.SocketAddress

//Construtores
InetSocketAddress (InetAddress addr, int port)
InetSocketAddress (int port)
InetSocketAddress (String hostName, int port)

// Métodos de Instância
public InetAddress getAddress ()
public String getHostName ()
public int getPort ()
public boolean isUnresolved ()
```

Exemplo 11.3

O programa na Tabela 11.5 mostra como podemos criar um endereço de *socket*. Perceba que o número da porta é separado do InetAddress por um caractere de dois pontos quando apresentado na tela.

Tabela 11.5 Esquema para o Exemplo 11.3.

```
1  import java.io.*;
2  public class SocketAddresses
3  {
4      public static void main (String [] args) throws IOException
5      {
6          InetAddress local = InetAddress.getLocalHost ();
7          int port = 65000;
8          InetSocketAddress sockAddr = new InetSocketAddress (local, port);
9          System.out.println (sockAddr);
10     }// Fim do main
11
12 }// Fim da classe
```

Resultado:

Behrouz/64.183.101.114:65000

11.1.2 Paradigma cliente-servidor

No Capítulo 2, discutimos o paradigma cliente-servidor. Dissemos que um *cliente* é um programa com tempo de vida finito que solicita serviços a um servidor. Também dissemos que um *servidor* é um programa com tempo de vida indeterminado que fornece serviços para os clientes. Um servidor no

paradigma cliente-servidor pode ser projetado como um servidor iterativo ou como um servidor concorrente. Um **servidor iterativo** trata uma a uma as requisições dos clientes. Enquanto o servidor estiver atendendo um cliente, os outros clientes precisam esperar. Isto é semelhante a uma pequena filial de um banco na qual há apenas um caixa para atender os clientes. Enquanto o caixa está atendendo um cliente, os outros clientes precisam fazer uma fila e esperar sua vez. Um **servidor concorrente** pode atender simultaneamente todos os clientes que seus recursos computacionais permitirem. A analogia irreal consiste em um grande banco com caixas suficientes para atender todos os clientes ao mesmo tempo. Quando um cliente chega ao banco, é direcionado a um caixa para ser atendido.

Conforme vimos no Capítulo 3, um protocolo da camada de transporte pode ser orientado à conexão ou não orientado à conexão. Essa distinção afeta a maneira como os programas cliente-servidor são projetados na camada de aplicação. Um par cliente-servidor que utiliza os serviços de uma camada de transporte não orientada à conexão, como o UDP, deve ser projetado como um par de programas não orientados à conexão. O cliente precisa entregar o seu pedido à camada de transporte na forma de um único bloco de dados e deve receber a resposta da camada de transporte na forma de um único bloco de dados. O mesmo procedimento deve ser seguido pelo servidor.

Um par cliente-servidor que utiliza os serviços de uma camada de transporte orientada à conexão, como o TCP, deve ser projetado como um par de programas orientados à conexão. O cliente deve entregar seu pedido à camada de transporte na forma de um fluxo de *bytes* e deve receber a resposta da camada de transporte na forma de um fluxo de *bytes*. O mesmo procedimento deve ser seguido pelo servidor.

Programas cliente-servidor

Conforme discutimos no Capítulo 2, as camadas abaixo da camada de aplicação na pilha de protocolos TCP/IP sabem como enviar pacotes para a Internet e como receber pacotes vindos da Internet. A situação na camada de aplicação é diferente. Temos dois grupos de programas cliente-servidor. O primeiro grupo consiste em programas cliente-servidor padrão, cujos programas são escritos e compilados para a linguagem de máquina do computador que estamos usando. O segundo grupo é composto por programas cliente-servidor que precisamos escrever para um propósito específico. Este capítulo é principalmente sobre esses tipos de programas, porém tentamos simular alguns programas-padrão para fins de aprendizado.

Interface de *sockets* em Java

No Capítulo 2 (Seção 5) discutimos que, embora existam algumas maneiras de escrever um aplicativo cliente ou servidor, nos concentramos na *interface socket*. Recomendamos a revisão dessa seção antes de nos concentrarmos na programação de redes usando Java.

11.2 PROGRAMANDO COM UPD

Para mantermos a coerência com a seção sobre programação de *sockets* do Capítulo 2, primeiramente discutiremos a programação de redes usando os serviços do UDP, um serviço não orientado à conexão. Começaremos discutindo sobre a abordagem iterativa e em seguida passaremos para a abordagem concorrente.

11.2.1 Abordagem iterativa

O UDP fornece um serviço não orientado à conexão e a comunicação é feita usando conjuntos de dados denominados datagramas do usuário. Na abordagem iterativa, o servidor trata um datagrama de usuário de cada vez. O restante dos datagramas de usuário que chegarem precisa esperar, não importando se eles estão vindo do mesmo cliente ou de outros.

Sockets usados pelo UDP

A Figura 11.1 mostra o tempo de vida dos *sockets* no cliente e no servidor A implementação em Java do UDP usa apenas um tipo de objeto da interface *socket*: as instâncias da classe *DatagramSocket*. Perceba que estamos discutindo uma comunicação iterativa, o que significa que quando um cliente está sendo atendido por um servidor, os outros clientes não podem ser atendidos (eles precisam esperar). Nesse tipo de comunicação, um cliente envia um datagrama de usuário, uma instância da classe *DatagramPacket* e recebe um datagrama de usuário.

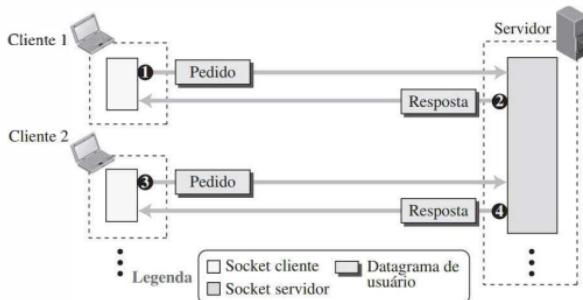



Figura 11.1 *Sockets* para comunicações UDP.

Classes

Embora o UDP use apenas um tipo de objeto da interface *socket*, também precisamos de objetos referentes à datagramas de usuário. Antes de escrevermos o código para um cliente e um servidor simples, mostraremos o resumo dos métodos fornecidos por essas duas classes.

Classe DatagramSocket

A classe DatagramSocket é usada para criar *sockets* no cliente e no servidor Ela também fornece métodos para enviar um datagrama de usuário, para receber um datagrama de usuário e para fechar o *socket*.

A Tabela 11.6 mostra as assinaturas de alguns dos métodos dessa classe.

Tabela 11.6 Alguns métodos da classe DatagramSocket.

```
public class java.net.DatagramSocket extends java.lang.Object

// Construtores
public DatagramSocket ()
public DatagramSocket (int localPort)
public DatagramSocket (int localPort, InetAddress localAddr)

// Métodos de Instância
public void send (DatagramPacket sendPacket)
public void receive (DatagramPacket recvPacket)
public void close ()
```

Classe DatagramPacket

A classe DatagramPacket é usada para criar pacotes de datagramas de usuário. A Tabela 11.7 mostra as assinaturas de alguns métodos dessa classe.

Tabela 11.7 Alguns métodos da classe DatagramPacket.

```
public final class java.net. DatagramPacket extends java.lang.Object

// Construtores
public DatagramPacket (byte [] data, int length)
public DatagramPacket (byte [] data, int length, InetAddress remoteAddr, int remotePort)

// Métodos de Instância
public InetAddress getAddress ()
public int getPort ()
public byte [] getData ()
public int getLength ()
```

Projeto do cliente UDP

A Figura 11.2 mostra o projeto dos objetos e seus relacionamentos no programa-cliente que vamos escrever. Antes de explicarmos esse projeto, é preciso mencionar que ele se aplica apenas aos nossos programas-cliente, que são muito simples.

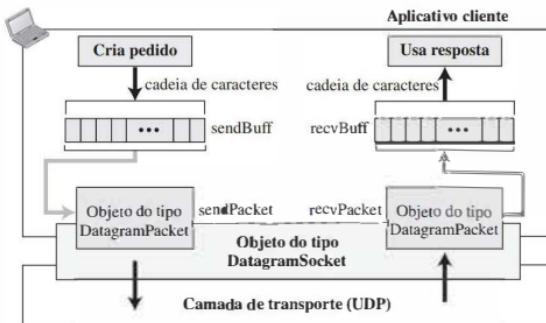



Figura 11.2 Projeto do cliente UDP.

Esse projeto mostra que temos um objeto do tipo *socket* (uma instância da classe DatagramSocket) cliente, criado pelo programa-cliente para fornecer uma conexão entre o cliente e a camada de transporte (UDP). Como o aplicativo fornece os dados em blocos, precisamos de um objeto da classe DatagramPacket para representar o pacote, de modo que o bloco de dados possa ser enviado para o *socket*. Precisamos também de um objeto do mesmo tipo para o pacote que contém o bloco de dados recebido do *socket*.

Os dados fornecidos aos pacotes de datagrama de usuário ou obtidos desses pacotes precisam estar na forma de *bytes*. Por isso, também mostramos dois vetores de *bytes*, denominados sendBuff e recvBuff, nos quais armazenamos esses *bytes* antes de passá-los para o datagrama de usuário. Por razões de simplicidade, nosso programa considera que temos um método que cria o pedido na forma de uma cadeia de caracteres (*string*), e um método que usa a resposta na forma de uma cadeia de caracteres. Nessa estrutura, precisamos converter o pedido em um vetor de *bytes* e converter o vetor de *bytes* na resposta.

Programa-cliente

A Tabela 11.8 mostra um programa-cliente simples que segue o projeto proposto na Figura 11.2. Construímos o programa inteiro como uma única classe, denominada UDPClient, com construtores e métodos de instância. As linhas 6 a 11 criam constantes e campos de dados (referências). Precisamos definir o tamanho dos *buffers* antes que possamos executar o programa. Explicamos brevemente o código após mostrarmos o programa.

Tabela 11.8 Um programa simples de cliente UDP

```
1   import java.net.*;
2   import java.io.*;
3
4   public class UDPClient
5   {
6       final int buffSize = ...;                       // Inserir um tamanho de buffer
7       DatagramSocket sock;                            // O socket
8       String request;                                 // Cadeia de caracteres do pedido
9       String response;                                // Cadeia de caracteres da resposta
10      InetAddress servAddr;                           // Endereço do servidor
11      int servPort;                                   // Porta do servidor
12
13      UDPClient (DatagramSocket s, String sName, int sPort)
14                                                      throws UnknownHostException
15      {
16          sock = s;
17          servAddr = InetAddress.getByName (sName);
18          servPort = sPort;
19      }
20
21      void makeRequest ()
22      {
23          // Insira aqui o código para criar a cadeia de caracteres do pedido.
24      }
```

(Continua)

Tabela 11.8 Um programa simples de cliente UDP. (*Continuação*)

```
25
26      void sendRequest ()
27      {
28              try
29              {
30                      byte [] sendBuff = new byte [buffSize];
31                      sendBuff = request.getBytes ();
32                      DatagramPacket sendPacket = new DatagramPacket (sendBuff,
33                                                      sendBuff.length, servAddr, servPort);
34                      sock.send(sendPacket);
35              }
36              catch (SocketException ex)
37              {
38                      System.err.println ("SocketException em getRequest");
39              }
40      }
41
42      void getResponse ()
43      {
44              try
45              {
46                      byte [] recvBuff = new byte [buffSize];
47                      DatagramPacket recvPacket = new DatagramPacket (recvBuff, buffSize);
48                      sock.receive (recvPacket);
49                      recvBuff = recvPacket.getData ();
50                      response = new String (recvBuff, 0, recvBuff.length);
51              }
52              catch (SocketException ex)
53              {
54                      System.err.println ("SocketException em getRequest");
55              }
56      }
57
58      void useResponse ()
59      {
```

(*Continua*)

Tabela 11.8 Um programa simples de cliente UDP (*Continuação*)

```
60              // Insira aqui o código para usar a cadeia de caracteres da resposta.
61          }
62
63          void close ()
64          {
65              sock.close ();
66          }
67
68          public static void main (String [] args) throws IOException, SocketException
69          {
70              final int servPort = ...;                    //Inserir o número de porta do servidor
71              final String servName = ...;                 // Inserir o nome do servidor
72              DatagramSocket sock = new DatagramSocket ();
73              UDPClient client = new UDPClient (sock, servName, servPort);
74              client.makeRequest ();
75              client.sendRequest ();
76              client.getResponse ();
77              client.useResponse ();
78              client.close ();
79          } // Fim do main
80      } // Fim da classe UDPClient
```

Método main

A execução do programa começa com o método main (linhas 68 a 79). O usuário precisa fornecer o número de porta do servidor (um inteiro) e o nome do servidor (uma cadeia de caracteres). O programa, então, cria uma instância da classe DatagramSocket (linha 72) e uma instância da classe UDPClient (linha 73) que é responsável por criar um pedido, enviar o pedido, receber a resposta, usar a resposta e fechar o *socket*. As linhas 74 a 78 invocam os métodos apropriados na classe UDPClient para realizar essas tarefas.

Métodos na classe UDPClient

A seguir, descrevemos os métodos na classe UDPClient.

Construtor O construtor (linhas 13 a 19) é muito simples. Ele recebe a referência para o *socket*, o nome e a porta do servidor. Ele usa o nome do servidor para localizar o endereço IP. Perceba que não é necessário fornecer a porta e o endereço IP do cliente, que são fornecidos pelo sistema operacional.

Método makeRequest Este método (linhas 21 a 24) encontra-se vazio em nosso projeto. Ele precisa ser preenchido pelo usuário do programa para criar a cadeia de caracteres do pedido. Mostramos alguns casos nos nossos exemplos.

Método sendRequest Este método (linhas 26 a 40) é responsável por diversas tarefas:

1. Cria um *buffer* de envio vazio (linha 30).
2. Preenche o *buffer* de envio com a cadeia de caracteres do pedido, criada pelo método makeRequest (linha 31).
3. Cria um pacote para o datagrama de usuário e o associa ao *buffer* de envio, ao endereço do servidor e à porta do servidor (linhas 32 e 33).
4. Envia o pacote usando o método *send* (enviar) definido na classe DatagramSocket (linha 34).

Método getResponse Este método (linhas 42 a 56) é responsável por diversas tarefas.

1. Cria um *buffer* de recepção vazio (linha 46).
2. Cria o datagrama de usuário de recepção e o associa ao *buffer* de recepção (linha 47).
3. Usa o método *receive* (receber) da classe DatagramSocket para receber a resposta do servidor e preenche o datagrama com o valor dessa resposta (linha 48).
4. Extrai os dados do pacote recebido e os armazena no *buffer* de recepção (linha 49).
5. Cria a cadeia de caracteres de resposta a ser usada pelo método useResponse (linha 50).

Método useResponse Este método (linhas 58 a 61) encontra-se vazio em nosso projeto. Ele deve ser preenchido pelo usuário do programa para processar a resposta enviada pelo servidor. Mostramos algumas possibilidades em nossos exemplos.

Método close Este método (linhas 63 a 66) fecha o *socket*.

Servidor UDP

A Figura 11.3 mostra o projeto dos objetos e seus relacionamentos no programa-servidor que vamos escrever. Antes de explicarmos esse projeto, é preciso mencionar novamente que ele se aplica apenas ao nosso programa-servidor, que é muito simples; ele não representa um projeto genérico, que pode ser mais complexo. Esse projeto mostra que temos um objeto do tipo DatagramSocket e dois objetos do tipo DatagramPacket.

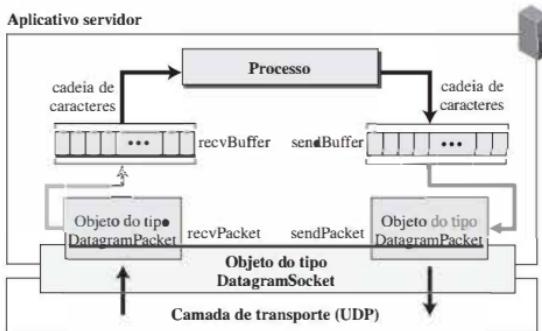



Figura 11.3 Projeto do servidor UDP.

Programa-servidor

A Tabela 11.9 mostra um programa-servidor simples que segue o projeto proposto na Figura 11.3. A seguir, fornecemos uma breve descrição dele.

Tabela 11.9 Um programa simples de servidor UDP.

```
1   import java.net.*;
2   import java.io.*;
3
4   public class UDPServer
5   {
6         final int buffSize = ...;                    // Inserir um tamanho de buffer
7         DatagramSocket sock;                         // O socket
8         String request;                              // Cadeia de caracteres do pedido
9         String response;                             // Cadeia de caracteres da resposta
10        InetAddress clientAddr;                      // Endereço do cliente
11        int clientPort;                              // Porta do cliente
12
13        UDPServer (DatagramSocket s)
14        {
15          sock = s;
16        }
17
18  void getRequest ()
19        {
20        try
21        {
22                    byte [] recvBuff = new byte [buffSize];
23                    DatagramPacket recvPacket = new DatagramPacket (recvBuff, buffSize);
24                    sock.receive (recvPacket);
25                    recvBuff = recvPacket.getData ();
26                    request = new String (recvBuff, 0, recvBuff.length);
27                    clientAddr = recvPacket.getAddress ();
28                    clientPort = recvPacket.getPort ();
29            }
30            catch (SocketException ex)
31            {
32                    System.err.println ("SocketException em getRequest");
```

(*Continua*)

Tabela 11.9 Um programa simples de servidor UDP. (*Continuação*)

```
33              }
34          catch (IOException ex)
35              {
36                  System.err.println ("IOException em getRequest");
37              }
38      }
39
40          void process ()
41      {
42          // Inserir código para processar o pedido e criar a resposta.
43      }
44
45      void sendResponse()
46      {
47          try
48          {
49              byte [] sendBuff = new byte [buffSize];
50              sendBuff = response.getBytes ();
51              DatagramPacket sendPacket = new DatagramPacket (sendBuff,
52                                          sendBuff.length, clientAddr, clientPort);
53              sock.send(sendPacket);
54          }
55          catch (SocketException ex)
56          {
57              System.err.println ("SocketException em sendResponse");
58          }
59          catch (IOException ex)
60          {
61              System.err.println ("IOException em sendResponse");
62          }
63      }
64
65      public static void main (String [] args) throws IOException, SocketException
66      {
67          final int port = ...;                       // Inserir o número de porta do servidor.
```

(*Continua*)

Tabela 11.9 Um programa simples de servidor UDP. *(Continuação)*

```
68            DatagramSocket sock = new DatagramSocket (port);
69            while (true)
70            {
71                UDPServer server = new UDPServer (sock);
72                server.getRequest ();
73                server.process ();
74                server.sendResponse ();
75            }
76        } // Fim do main
77    } // Fim da classe UDPServer
```

Método main

A execução do programa começa com o método main (linhas 65 a 76). O usuário precisa fornecer o número de porta do servidor (um inteiro) no qual o programa-servidor está sendo executado (linha 67). Não é necessário fornecer o endereço ou o nome do *host*; essas informações são fornecidas pelo sistema operacional. O programa, então, cria uma instância da classe DatagramSocket (linha 68) utilizando o número de porta especificado. O programa executa um laço infinito (linha 69), de forma que cada cliente é atendido em uma iteração do laço por meio da criação de uma nova instância da classe UDPServer e da chamada de seus três métodos de instância.

Métodos na classe UDPServer

A seguir, descrevemos os métodos nessa classe.

Construtor O construtor (linhas 13 a 16) é muito simples. Ele recebe a referência para o *socket* cliente e a armazena na variável *sock*.

Método getRequest Este método é responsável por diversas tarefas (linhas 18 a 38):

1. Cria um *buffer* de recepção (linha 22).
2. Cria o pacote com o datagrama de usuário e o associa ao *buffer* (linha 23).
3. Recebe o conteúdo do datagrama de usuário (linha 24).
4. Extrai os dados do datagrama de usuário e os armazena no *buffer* (linha 25).
5. Converte os *bytes* do *buffer* de recepção na cadeia de caracteres que compõe o pedido (linha 26).
6. Extrai o endereço IP do cliente que enviou o pacote (linha 27).
7. Extrai o número da porta do cliente que enviou o pacote (linha 28).

Método process Este método (linhas 40 a 43) encontra-se vazio em nosso projeto. Ele precisa ser preenchido pelo usuário do programa de modo a processar o pedido e criar uma resposta. Mostramos algumas possibilidades em nossos exemplos.

Método sendResponse Este método (linhas 45 a 63) é responsável por diversas tarefas.

1. Cria um *buffer* de envio vazio (linha 49).
2. Transforma a cadeia de caracteres de resposta em *bytes* e os armazena no *buffer* de envio (linha 50).

3. Cria um novo datagrama de usuário e o preenche com os dados presentes no *buffer* (linhas 51 e 52).
4. Envia o pacote que contém o datagrama de usuário criado (linha 53).

Exemplo 11.4

O exemplo mais simples possível consiste em simular o aplicativo cliente/servidor padrão de *echo*. Esse programa é usado para verificar se um servidor está ativo. Uma mensagem curta é enviada pelo cliente. A mensagem é ecoada de volta exatamente como foi recebida. Embora o aplicativo-padrão use a porta bem conhecida 7, usamos o número de porta 52007 para simular o aplicativo no servidor.

1. No programa-cliente, fixamos a porta do servidor em 52007 e o nome do servidor como o nome do computador ou o endereço do computador ("x.y.z.t") no qual o programa-servidor está sendo executado. Também modificamos os métodos makeRequest e useResponse conforme mostrado a seguir:

```
void makeRequest ()
{
            request = "Ola";
}
```

```
void useResponse ()
{
            System.out.println (response);
}
```

2. No programa-servidor, fixamos a porta do servidor em 52007. Também modificamos o método *process* no programa-servidor conforme mostrado a seguir:

```
void process ()
{
            response = request;
}
```

3. Deixamos o programa-servidor executando em um *host* e, em seguida, executamos o programa-cliente em outro *host*. Podemos executar ambos no mesmo *host* se executarmos o programa-servidor em segundo plano.

Exemplo 11.5

Neste exemplo, transformamos nosso servidor em um servidor simples de data/hora. Ele devolve a data e a hora no local onde o servidor está sendo executado.

1. No programa-cliente, fixamos a porta do servidor em 40013 e o nome do servidor como o nome do computador ou o endereço do computador ("x.y.z.t") no qual o programa-servidor está sendo executado. Também modificamos os métodos makeRequest e useResponse usando o seguinte código:

```
void makeRequest ()
{
            request = "Envie a data e a hora, por favor.";
}
```

```
void use Response ()
{
            System.out.println (response);
}
```

2. No programa-servidor (Tabela 11.9), adicionamos uma declaração no inicio do programa para podermos usar as classes Calendar e Date (import java.util. *;). Fixamos a porta do servidor em 40013. Também alteramos o método process no programa servidor para

```
void process ()
{
		Date date = Calendar.getInstance ().getTime ();
		response = date.toString ();
}
```

O método process usa a classe calendar para obter a data e hora atuais e, em seguida, transforma a data em uma cadeia de caracteres para ser armazenada na variável *response* (contendo a resposta).

3. Deixamos o programa servidor executando em um *host* e, em seguida, executamos o programa-cliente em outro *host*. Podemos executar ambos no mesmo *host* se executarmos o programa-servidor em segundo plano.

Exemplo 11.6

Neste exemplo, precisamos usar o nosso programa cliente-servidor simples para medir o tempo (em milissegundos) que leva para enviar uma mensagem do cliente para o servidor.

1. No programa-cliente, adicionamos uma declaração no início do programa para podermos usar a classe Date (import java.util. *;). Também fixamos a porta do servidor em 40013 e o nome do servidor como o nome do computador ou o endereço do computador ("x.y.z.t") no qual o programa-servidor está sendo executado. Também modificamos os métodos makeRequest e useResponse usando o seguinte código:

```
void makeRequest ()
{
	Date date = new Date ();
	long time = date.getTime ();
	request = String.valueOf (time);
}
```

```
void useResponse () {
	Data date = new Date ();
	long now = date.getTime ();
	long elapsedTime = now – Long.parse(response));
	System.out.println ("Tempo decorrido = " + elapsedTime + " milisegundos.";
}
```

Observe que, embora não tenhamos que enviar o valor do instante de tempo para o servidor, fazemos isso para não alterar a estrutura do nosso programa-cliente.

2. No programa-servidor, fixamos a porta do servidor em 40013. Também substituímos o método *process* no programa-servidor por

```
void process ()
{
		response = request;
}
```

3. Deixamos o programa-servidor executando em um *host* e, em seguida, executamos o programa-cliente em outro *host*. Podemos executar ambos no mesmo *host* se executarmos o programa-servidor em segundo plano.

11.2.2 Abordagem concorrente

A abordagem iterativa para o programa-servidor UDP é boa o suficiente para a maioria das aplicações porque, após processar e enviar um datagrama de usuário, o servidor está pronto para servir outros clientes. Entretanto, se o processamento de um datagrama de usuário demorar muito tempo, um cliente pode monopolizar o servidor. Os programas-servidor concorrentes foram projetados para resolver esse problema usando *threads**.

Programa-servidor

A Tabela 11.10 mostra um programa-servidor concorrente. Ele é quase idêntico à sua versão iterativa (Tabela 11.9), exceto pelo fato de que a classe do servidor implementa a interface Runnable. Isto nos permite substituir (*override*) o método run() da classe e incluir nesse método todas as chamadas que anteriormente se encontravam no método main (linhas 18 a 23). No método main, criamos uma instância da classe Thread e associamos o objeto *thread* a uma instância da classe que representa o servidor. Nesse caso, cada cliente é atendido em uma *thread* distinta. Comparando essa versão com a solução iterativa (Tabela 11.9), vemos que os três métodos principais (request, process e response) são executados no método run e não no método main.

Tabela 11.10 Um programa simples de servidor UDP concorrente.

```
1   import java.net.*;
2   import java.io.*;
3
4   public class ConcurUDPServer implements Runnable
5   {
6       final int buffSize = ...;           // Inserir um tamanho de buffer aqui.
7       DatagramSocket servSock;            // O socket
8       String request;                     // Cadeia de caracteres do pedido
9       String response;                    // Cadeia de caracteres da resposta
10      InetAddress clientAddr;             // Endereço do cliente
11       int clientPort; // Porta do cliente
12
13       ConcurUDPServer (DatagramSocket s)
14       {
15           servSock = s;
16       }
17
18       public void run ()
19       {
```

(*Continua*)

* N. de T.: Uma *thread* (também conhecida como linha de execução ou fluxo de execução) consiste em um processo que é executado independentemente de outros processos, possivelmente em paralelo a eles (caso haja suporte do *hardware*).

Tabela 11.10 Um programa simples de servidor UDP concorrente. (*Continuação*)

```
20            getRequest ();
21            process ();
22            sendResponse ();
23        }
24
25        void getRequest ()
26        {
27            try
28            {
29                byte [] recvBuff = new byte [buffSize];
30                DatagramPacket recvPacket = new DatagramPacket (recvBuff, buffSize);
31                sock.receive (recvPacket);
32                recvBuff = recvPacket.getData ();
33                request = new String (recvBuff, 0, recvBuff.length);
34                clientAddr = recvPacket.getAddress ();
35                clientPort = recvPacket.getPort ();
36            }
37            catch (SocketException ex)
38            {
39                System.err.println ("SocketException em getRequest");
40            }
41            catch (IOException ex)
42            {
43                System.err.println ("IOException em getRequest");
44            }
45        }
46
47        void process ()
48        {
49            // Inserir o código para este processo.
50        }
51
52        void sendResponse ()
53        {
54            try
```

(*Continua*)

Tabela 11.10 Um programa simples de servidor UDP concorrente. (*Continuação*)

```
55              {
56                  byte [] sendBuff = new byte [buffSize];
57                  sendBuff = response.getBytes ();
58                  DatagramPacket sendPacket = new DatagramPacket (sendBuff,
59                                                   sendBuff.length, clientAddr, clientPort);
60                  sock.send(sendPacket);
61              }
62              catch (SocketException ex)
63              {
64                  System.err.println ("SocketException em sendResponse");
65              }
66              catch (IOException ex)
67              {
68                  System.err.println ("IOException em sendResponse");
69              }
70          }
71
72          public static void main (String [] args) throws IOException, SocketException
73          {
74              final int port = ...;                    // Inserir o número de porta
75              DatagramSocket sock = new DatagramSocket (port);
76              while (true)
77              {
78                  ConcurUDPServer server = new ConcurUDPServer (sock);
79                  Thread thread = new Thread (server);
80                  thread.start ();
81              }
82          } // Fim do main
83
84      } // Fim da classe UDPServer
```

Exemplo 11.7

Repetimos o Exemplo 11.5 usando a abordagem concorrente. Precisamos de diversos computadores para, simultaneamente, enviar o pedido e receber a resposta.

Exemplo 11.8

Repetimos o Exemplo 11.6 usando a abordagem concorrente. Precisamos de diversos computadores para, simultaneamente, enviar o pedido e receber a resposta.

11.3 PROGRAMANDO COM TCP

Agora, estamos prontos para discutir a programação de redes usando o serviço do TCP, um serviço orientado à conexão. Primeiramente, discutimos como escrever um programa-cliente e um programa-servidor usando a abordagem iterativa. Em seguida, mostramos como podemos modificar o programa-servidor para transformá-lo em um programa concorrente.

11.3.1 Abordagem iterativa

Embora a abordagem iterativa seja raramente usada na programação TCP, ela é a base da abordagem concorrente. Nesta última abordagem, um servidor atende os clientes um a um. Quando o servidor começa a atender um cliente, os outros clientes precisam esperar.

Dois tipos de *sockets*

A implementação do TCP em Java usa duas classes de objetos do tipo *socket*: ServerSocket e Socket. Conforme vimos no Capítulo 3, a comunicação usando TCP ocorre em três fases: estabelecimento da conexão, transferência de dados e encerramento da conexão. Um cliente usa somente um objeto da classe Socket; já um servidor usa um objeto da classe ServerSocket durante o estabelecimento de conexão e um objeto da classe Socket durante as outras duas fases. A Figura 11.4 mostra o tempo de vida dos *sockets* tanto nos clientes como no servidor. Perceba que estamos discutindo uma comunicação iterativa, o que significa que, enquanto um cliente estiver conectado a um servidor, outros clientes não podem se conectar (eles precisam aguardar).

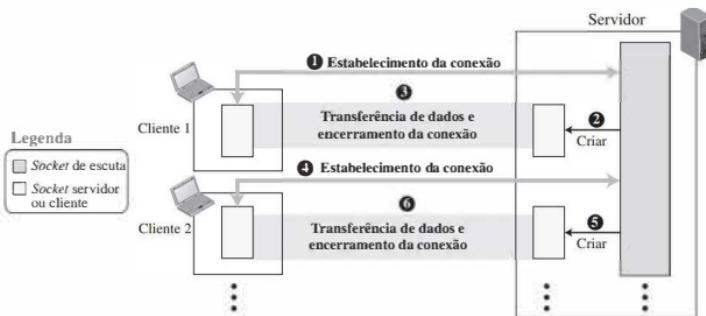



Figura 11.4 Objetos das classes ServerSocket e Socket na comunicação TCP.

A lógica por trás de haver duas classes do tipo *socket* é separar a fase de estabelecimento da conexão da fase de transferência de dados. Na comunicação iterativa, basta dizer que o ServerSocket, que também é chamado *socket passivo* ou *socket de escuta*, é responsável apenas pelo estabelecimento de uma conexão. Depois da conexão ser estabelecida com um cliente, o objeto ServerSocket cria um objeto Socket para atender o cliente e espera até que o cliente encerre a conexão.

Classes

Antes de escrever o código para um cliente e um servidor simples, mostraremos o resumo dos métodos utilizados nas duas classes discutidas anteriormente.

Classe ServerSocket

A classe ServerSocket é usada para criar os *sockets* de escuta utilizados no estabelecimento da comunicação no TCP (*handshaking*). A Tabela 11.11 mostra as assinaturas de alguns dos métodos dessa classe. A variável *backlog* determina o número de solicitações de conexão que podem ser enfileiradas, esperando para serem atendidas.

Tabela 11.11 Resumo da classe ServerSocket.

```
public class java.net. ServerSocket extends java.lang.Object

// Construtores
ServerSocket ()
ServerSocket (int localPort)
ServerSocket (int localPort, int backlog)
ServerSocket (int localPort, int backlog, InetAddress bindAddr)

// Métodos de instância
public Socket accept ()
public void bind (int localPort, int backlog)
public InetAddress getInetAddress ()
public SocketAddress getLocalSocketAddress ()
```

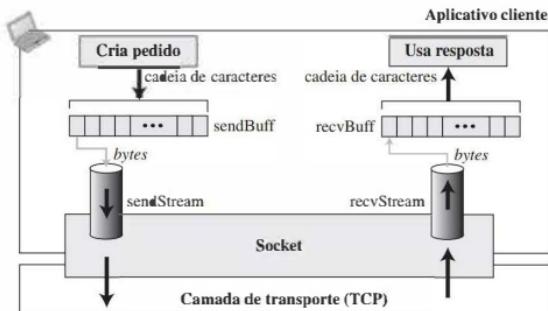
Classe Socket

A classe Socket é usada no TCP para a transferência de dados. A Tabela 11.12 mostra as assinaturas de alguns dos métodos dessa classe.

Tabela 11.12 Resumo da classe Socket.

```
public class java.net. Socket extends java.lang.Object

// Construtores
Socket ()
Socket (String remoteHost, int remotePort)
Socket (InetAddress remoteAddr, int remotePort)
Socket (String remoteHost, int remotePort, InetAddress localAddr, int localPort)
Socket (InetAddress remoteAddr, int remotePort, InetAddress localAddr, int localPort)
// Métodos de instância
public void connect (SocketAddress destination)
public void connect (SocketAddress destination, int timeout)
public InetAddress getInetAddress ()
public int getPort ()
```

(*Continua*)

Tabela 11.12 Resumo da classe Socket. *(Continuação)*
public InetAddress **getLocalAddress** ()
public int **getLocalPort** ()
public SocketAddress **getRemoteSocketAddress** ()
public SocketAddress **getLocalSocketAddress** ()
public InputStream **getInputStream** ()
public OutputStream **getOutputStream** ()
public void **shutdownInput** ()
public void **shutdownOutput** ()
public void **close** ()

Projeto do cliente TCP

A Figura 11.5 mostra o projeto dos objetos e seus relacionamentos no programa-cliente que descreveremos. Antes de explicarmos esse projeto, é preciso mencionar que ele se aplica apenas ao nosso programa-cliente, que é muito simples; ele não representa um projeto para aplicações-padrão, como aplicações HTTP, que podem ser mais complexas.

Figura 11.5 Projeto do cliente TCP.

O projeto mostra que temos um objeto do tipo *socket* (pertencente à classe Socket) criado pelo programa-cliente para fornecer uma conexão entre o cliente e a camada de transporte (TCP). Precisamos de um fluxo de *bytes* para enviar *bytes* para o *socket*. Precisamos também de um fluxo de *bytes* para receber *bytes* do *socket*. Em Java, esses dois fluxos podem ser criados usando os métodos getOutputStream e getInputStream, fornecidos pela classe Socket (Tabela 11.12). No nosso programa, temos também dois vetores de *bytes*, denominados sendBuff e recvBuff, para armazenar *bytes* antes de enviá-los para o objeto sendStream e para armazenar *bytes* depois de recebê-los do objeto recvStream. Para simplificar, nosso programa assume que temos um método que cria a solicitação na forma de uma cadeia de caracteres (*string*), e um método que usa a resposta na forma de uma cadeia de caracteres. Nesse caso, precisamos converter o pedido, transformando a cadeia de caracteres em um vetor de *bytes* e transformando o vetor de *bytes* na cadeia de caracteres de resposta. Entretanto, podemos utilizar estratégias diferentes em outros projetos.

Programa-cliente TCP

A Tabela 11.13 mostra um programa-cliente simples que segue o projeto da Figura 11.5. Construímos o programa inteiro como uma única classe, denominada TCPClient, com construtores e métodos de instância. Explicamos brevemente o código a seguir.

Tabela 11.13 Um programa simples de cliente TCP.

```
1   import java.net.*;
2   import java.io.*;
3
4   public class TCPClient
5   {
6           Socket sock; // O socket
7           OutputStream sendStream;                        // Fluxo de envio
8           InputStream recvStream;                         // Fluxo de recepção
9           String request;                                 // Pedido
10          String response;                                // Resposta
11
12          TCPClient (String server, int port) throws IOException, UnknownHostException
13          {
14                  sock = new Socket (server, port);
15                  sendStream = sock.getOutputStream ();
16                  recvStream = sock.getInputStream ();
17          }
18
19          void makeRequest ()
20          {
21                  // Insira aqui o código para criar a cadeia de caracteres do pedido.
22          }
23
24          void sendRequest ()
25          {
26                try
27                {
28                        byte [] sendBuff = new byte [request.length ()];
29                        sendBuff = request.getBytes ();
30                        sendStream.write (sendBuff, 0, sendBuff.length);
31                }
```

(*Continua*)

Tabela 11.13 Um programa simples de cliente TCP. (*Continuação*)

```
32          catch (IOException ex)
33          {
34              System.err.println ("IOException em sendRequest");
35          }
36      }
37
38      void getResponse ()
39      {
40          try
41          {
42              int dataSize;
43              while ((dataSize = recvStream.available ()) == 0);
44              byte [] recvBuff = new byte [dataSize];
45              recvStream.read (recvBuff, 0, dataSize);
46              response = new String (recvBuff, 0, dataSize);
47          }
48          catch (IOException ex)
49          {
50              System.err.println ("IOException em getResponse");
51          }
52      }
53
54      void useResponse ()
55      {
56          // Insira aqui o código para usar a cadeia de caracteres da resposta.
57      }
58
59      void close ()
60      {
61          try
62          {
63              sendStream.close ();
64              recvStream.close ();
65              sock.close ();
66          }
```

(*Continua*)

Tabela 11.13 Um programa simples de cliente TCP. *(Continuação)*

67	catch (IOException ex)
68	{
69	System.err.println ("IOException em close");
70	}
71	}
72	
73	public static void main (String [] args) throws IOException
74	{
75	final int servPort = ...; // Fornecer a porta do servidor
76	final String servName = "..."; // Fornecer o nome do servidor
77	TCPClient client = new TCPClient (servName, servPort);
78	client.makeRequest ();
79	client.sendRequest ();
80	client.getResponse ();
81	client.useResponse ();
82	client.close ();
83	}// Fim do main
84	}// Fim da classe TCPClient

Método main

A execução do programa se inicia com o método main (linhas 73 a 83). O usuário precisa fornecer o número de porta do servidor (um inteiro) e o nome do servidor (uma cadeia de caracteres). O programa, então, cria uma instância da classe TCPClient (linha 77) responsável por criar um pedido, enviar o pedido, receber a resposta, usar a resposta e fechar o *socket* e os fluxos correspondentes. As linhas 78 a 82 invocam os métodos apropriados da classe TCPClient para realizar essas tarefas.

Métodos na classe TCPClient

A seguir, descrevemos os métodos na classe TCPClient.

Construtor O construtor (linhas 13 a 19) é muito simples. Ele recebe a referência para o nome e para a porta do servidor. Ele usa o nome do servidor para localizar o endereço IP do servidor. O construtor também cria os fluxos de entrada e de saída para enviar e receber dados.

Método makeRequest Este método (linhas 19 a 22) encontra-se vazio em nosso projeto. Ele precisa ser preenchido pelo usuário do programa para criar a cadeia de caracteres do pedido. Mostramos alguns casos nos nossos exemplos.

Método sendRequest Este método (linhas 24 a 36) é responsável por diversas tarefas:

1. Cria um *buffer* de envio vazio (linha 28).
2. Preenche o *buffer* de envio com a cadeia de caracteres do pedido, criada pelo método makeRequest (linha 29).
3. Escreve o conteúdo do *buffer* de envio, utilizando o método write do fluxo de saída (linha 30).

Método getResponse Este método (linhas 38 a 53) é responsável por diversas tarefas.

1. Procura continuamente por *bytes* disponíveis e verifica seu tamanho (linha 43) em um laço vazio.
2. Cria um *buffer* com o tamanho apropriado (linha 44).
3. Lê os dados do fluxo de entrada e os armazena no *buffer* de recepção (linha 45).
4. Converte os *bytes* do *buffer* de recepção na cadeia de caracteres de resposta a ser usada pelo método useResponse (linha 46).

Método useResponse Este método (linhas 54 a 57) encontra-se vazio em nosso projeto. Ele deve ser preenchido pelo usuário do programa para processar a resposta enviada pelo servidor. Mostramos alguns casos em nossos exemplos.

Método close Este método (linhas 59 a 71) primeiro fecha os fluxos de entrada e de saída e depois fecha o *socket*.

Servidor TCP

A Figura 11.6 mostra o projeto dos objetos e seus relacionamentos no programa-servidor que vamos escrever. Antes de explicarmos esse projeto, é preciso mencionar que ele se aplica apenas ao nosso programa-servidor, que é muito simples; ele não representa um projeto genérico, que pode ser mais complexo.

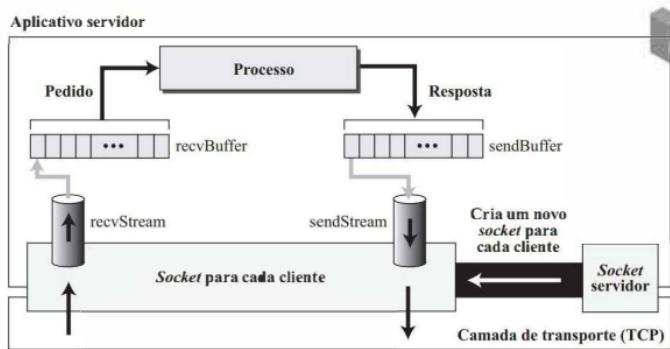



Figura 11.6 Projeto do servidor TCP para cada conexão de cliente.

O projeto mostra que temos um *socket* de escuta (uma instância de ServerSocket) criado pelo programa-servidor, o qual fica esperando os pedidos de conexão dos clientes. O *socket* de escuta cria iterativamente um *socket* (instância da classe Socket) para cada cliente usando o método accept definido na classe ServerSocket.

Programa-servidor

A Tabela 11.14 mostra um programa-servidor simples que segue o projeto da Figura 11.6. A seguir, damos uma breve descrição do programa. Consideramos que o pedido e a resposta consistem em pequenos blocos de dados.

Tabela 11.14 Um programa simples de servidor TCP.

```
1  import java.net.*;
2  import java.io.*;
3
4  public class TCPServer
5  {
6      Socket sock; // O socket
7      InputStream recvStream;                 // Fluxo de recepção
8      OutputStream sendStream;                // Fluxo de envio
9      String request;                         // Pedido
10     String response;                        // Resposta
11
12     TCPServer (Socket s) throws IOException, UnknownHostException
13     {
14         sock = s;
15         recvStream = sock.getInputStream ();
16         sendStream = sock.getOutputStream ();
17     }
18
19     void getRequest ()
20     {
21         try
22         {
23             int dataSize;
24             while ((dataSize = recvStream.available ()) == 0);
25             byte [] recvBuff = new byte [dataSize];
26             recvStream.read (recvBuff, 0, dataSize);
27           request = new String (recvBuff, 0, dataSize);
28         }
29         catch (IOException ex)
30         {
31           System.err.println ("IOException em getRequest");
32         }
33     }
34
35     void process()
```

(*Continua*)

Tabela 11.14 Um programa simples de servidor TCP. (*Continuação*)

```
36      {
37          // Inserir o código para processar o pedido e criar a resposta.
38      }
39
40      void sendResponse ()
41      {
42          try
43          {
44              byte [] sendBuff = new byte [response.length ()];
45              sendBuff = response.getBytes ();
46              sendStream.write (sendBuff, 0, sendBuff.length);
47          }
48          catch (IOException ex)
49          {
50              System.err.println ("IOException em sendResponse");
51          }
52      }
53
54      void close ()
55      {
56          try
57          {
58              recvStream.close ();
59              sendStream.close ();
60                sock.close ();
61            }
62            catch (IOException ex)
63            {
64                System.err.println ("IOException em close");
65            }
66      }
67
68      public static void main (String [] args) throws IOException
69      {
70            final int port = ...;                          // Fornecer o número de porta
```

(*Continua*)

Tabela 11.14 Um programa simples de servidor TCP (*Continuação*)

71	ServerSocket listenSock = new ServerSocket (port);
72	while (true)
73	{
74	TCPServer server = new TCPServer (listenSock.accept ());
75	server.getRequest ();
76	server.process ();
77	server.sendResponse ();
78	server.close ();
79	}
80	} // Fim do main
81	}// Fim da classe TCPServer

Método main

A execução do programa se inicia com o método main (linha 68). O usuário precisa fornecer o número de porta do servidor (um inteiro) no qual o programa-servidor está sendo executado (linha 70). Não é necessário fornecer o endereço ou o nome do *host*; essas informações são fornecidas pelo sistema operacional. O programa, então, cria uma instância da classe ServerSocket (linha 71) utilizando o número de porta especificado. O programa executa um laço infinito (linha 72), de forma que cada cliente é atendido em uma iteração do laço por meio da criação de uma nova instância da classe TCPServer e da chamada de seus quatro métodos de instância.

Métodos na classe TCPServer

A seguir, descrevemos os métodos nessa classe.

Construtor O construtor (linhas 12 a 17) é muito simples. Ele recebe a referência para o *socket* e também cria os fluxos de envio e de recepção.

Método getRequest Este método (linhas 19 a 33) é responsável por diversas tarefas:

1. Procura continuamente por *bytes* disponíveis e verifica seu tamanho (linha 24).
2. Cria um *buffer* com o tamanho apropriado (linha 25).
3. Lê os dados do fluxo de entrada e os armazena no *buffer* de recepção (linha 26).
4. Converte os *bytes* do *buffer* de recepção na cadeia de caracteres de resposta a ser usada pelo método process (linha 27).

Método process Este método (linhas 35 a 38) encontra-se vazio em nosso projeto. Ele precisa ser preenchido pelo usuário do programa de modo a processar o pedido e criar uma resposta. Mostramos alguns casos em nossos exemplos.

Método sendResponse Este método (linhas 40 a 52) é responsável por diversas tarefas.

1. Cria um *buffer* de envio vazio (linha 44).
2. Transforma a cadeia de caracteres de resposta em *bytes* e os coloca no *buffer* de envio (linha 45).
3. Escreve os *bytes* no objeto sendStream (linha 46).

Método close Este método (linhas 54 a 66) é responsável por fechar os fluxos de entrada e saída criados para cada cliente.

11.3.2 Abordagem concorrente

A abordagem iterativa para o programa-servidor TCP pode permitir que um cliente monopolize um servidor, impedindo que ele dê atenção aos pedidos de outros clientes. Os programas-servidor concorrentes foram criados para resolver esse problema. No passado, quando os servidores concorrentes eram escritos na linguagem C no ambiente UNIX, eram usados múltiplos processos para servir múltiplos clientes. Em Java, essa tarefa é executada por meio de diversas *threads*.

Programa-servidor

A Tabela 11.15 mostra um programa-servidor concorrente. Ele é quase idêntico à sua versão iterativa, exceto pelo fato de que a classe que representa o servidor implementa a interface Runnable. Isto nos permite substituir (*override*) o método run() dessa classe e incluir nesse método todas as chamadas que anteriormente se encontravam no método main (linhas 19 a 26). No método main, criamos uma instância da classe Thread e associamos o objeto *thread* a uma instância da classe que representa o servidor. Nesse caso, cada cliente é atendido em uma *thread* distinta.

Tabela 11.15 Um programa simples de servidor TCP concorrente.

```
1   import java.net.*;
2   import java.io.*;
3
4   public class ConcurTCPServer implements Runnable
5   {
6       Socket sock;                                // O socket
7       InputStream recvStream;                     // Fluxo de recepção
8       OutputStream sendStream;                    // Fluxo de envio
9       String request;                             // Pedido
10      String response;                            // Resposta
11
12      ConcurTCPServer (Socket s) throws IOException
13      {
14          sock = s;
15          recvStream = sock.getInputStream ();
16          sendStream = sock.getOutputStream ();
17      }
18
19      public void run()
```

(Continua)

Tabela 11.15 Um programa simples de servidor TCP concorrente. (*Continuação*)

```
20      {
21          getRequest ();
22          process ();
23          sendResponse ();
24          close();
25      }
26
27      void getRequest ()
28      {
29          try
30          {
31              int dataSize;
32              while ((dataSize = recvStream.available ()) == 0);
33              byte [] recvBuff = new byte [dataSize];
34              recvStream.read (recvBuff, 0, dataSize);
35              request = new String (recvBuff, 0, dataSize);
36          }
37          catch (IOException ex)
38          {
39              System.err.println ("IOException em getRequest");
40          }
41      }
42
43      void process ()
44      {
45          // Inserir código para processar pedido e criar a resposta.
46      }
47
48      void sendResponse ()
49      {
50          try
51          {
52              byte [] sendBuff = new byte [response.length ()];
53              sendBuff = response.getBytes ();
```

(*Continua*)

Tabela 11.15 Um programa simples de servidor TCP concorrente. (*Continuação*)

```
54                  sendStream.write (sendBuff, 0, sendBuff.length);
55              }
56              catch (IOException ex)
57              {
58                  System.err.println ("IOException em sendResponse");
59              }
60          }
61
62          void close ()
63          {
64              try
65              {
66                  recvStream.close ();
67                  sendStream.close ();
68                  sock.close ();
69              }
70              catch (IOException ex)
71              {
72                  System.err.println ("IOException em close");
73              }
74          }
75
76          public static void main (String [] args) throws IOException
77          {
78              final int port = ...;                          // Fornecer a porta do servidor
79              ServerSocket listenSock = new ServerSocket (port);
80              while (true)
81              {
82                  ConcurTCPServer server = new ConcurTCPServer (listenSock.accept ());
83                  Thread thread = new Thread (server);
84                  thread.start ();
85              }
86          } // Fim do main
87      } // Fim da classe ConcurTCPServer
```

11.4 MATERIAL DO FINAL DO CAPÍTULO

11.4.1 Leitura adicional

Diversos livros fornecem uma cobertura bastante completa de programação de redes usando a linguagem Java, incluindo: [Cal & Don 08], [Pit 06] e [Har 05].

11.4.2 Termos-chave

- servidor concorrente
- servidor iterativo

11.4.3 Resumo

A programação de redes definitivamente precisa lidar com endereços IP e números de porta. Em Java, um endereço IP é uma instância da classe InetAddress. As duas subclasses Inet4Address e Inet6Address também são definidas para representar explicitamente endereços IPv4 e IPv6. Um número de porta é representado como um inteiro. Em Java, um endereço de *socket*, que consiste em uma combinação de um endereço IP e de um número de porta, é representado pela classe SocketAddress; porém, o InetSocketAddress, uma subclasse da classe SocketAddress, é normalmente utilizado em programação Java.

No paradigma cliente-servidor, a comunicação se dá entre dois programas da camada de aplicação, um cliente e um servidor. Um cliente é um programa com tempo de vida finito que solicita serviços a um servidor. Um servidor, no paradigma cliente-servidor, pode ser projetado como um servidor iterativo ou como um servidor concorrente. Um servidor iterativo trata uma a uma as requisições dos clientes. Um servidor concorrente pode atender simultaneamente todos os clientes que seus recursos computacionais permitirem. Um par cliente-servidor que usa os serviços de uma camada de transporte não orientada à conexão, como o UDP, deve ser projetado como um par de programas não orientados à conexão. Um par cliente-servidor que utiliza os serviços de uma camada de transporte orientada à conexão, como o TCP, deve ser projetado como um par de programas orientados à conexão.

Embora existam algumas maneiras de escrever um aplicativo-cliente ou servidor, discutimos apenas a abordagem baseada na interface *socket*. A ideia é criar uma nova camada de abstração, a camada da interface *socket*, entre o sistema operacional e a camada de aplicação.

A implementação em Java de aplicativos baseados em UDP usa duas classes: DatagramSocket e DatagramPacket. A primeira é usada para criar um objeto do tipo *socket*; a segunda é usada para criar os datagramas de usuário transmitidos. A programação em Java, quando implementamos aplicativos baseados em TCP, usa duas classes: ServerSocket e Socket. A primeira é usada apenas durante o estabelecimento da conexão; a segunda é usada no restante da comunicação. A abordagem concorrente na programação de redes usando Java emprega múltiplas *threads*, permitindo que o servidor atenda cada cliente em uma *thread* distinta.

11.5 ATIVIDADES PRÁTICAS

11.5.1 Testes

Diversos testes interativos relativos a este capítulo podem ser encontrados no *site* www.mhhe.com/forouzan. Recomendamos que o estudante realize os testes para verificar o seu entendimento dos materiais antes de continuar com as atividades práticas.

Questões

11-1 Como um endereço IP é representado em Java?

11-2 Como o Java diferencia endereços IPv4 de endereços IPv6?

11-3 Um número de porta na pilha de protocolos TCP/IP consiste em um número inteiro de 16 *bits* sem sinal. Como podemos representar um número de porta em Java usando um número inteiro de 32 *bits*?

11-4 Por que você acha que o Java usa uma instância de classe e não simplesmente um número inteiro para representar um endereço IP?

11-5 Por que você acha que o Java não fornece construtores para a classe InetAddress?

11-6 Você sabe que o nome de domínio de um computador é "umNegocio.com". Escreva um código em Java para criar um objeto da classe InetAddress associado a esse computador.

11-7 Você sabe que o endereço IP de um computador é "23.14.76.44". Escreva um código em Java para criar um objeto da classe InetAddress associado a esse endereço.

11-8 Você acredita que um computador com o nome de domínio "umaUniversidade.edu" tem vários endereços IP. Escreva um código em Java para criar um vetor de objetos da classe InetAddress associados a esse computador.

11-9 Você acredita que um computador cujo endereço IP é "14.26.89.101" tem mais endereços IP. Escreva um código em Java para criar um vetor de objetos da classe InetAddress associados a esse computador.

11-10 Você deseja escrever um programa no qual precisa criar uma referência ao InetAddress do computador no qual está trabalhando. Escreva um código em Java para criar o objeto correspondente.

11-11 Você deseja criar todos os objetos InetAddress associados com o computador no qual está trabalhando. Escreva o código em Java para fazer isso.

11-12 Escreva um código em Java que armazena o número de porta 62230 em uma variável em Java e garanta que o número seja armazenado como um inteiro sem sinal.

11-13 Um endereço de *socket* é a combinação de um endereço IP e um número de porta que define um aplicativo sendo executado em um *host*. Podemos criar uma instância da classe InetSocketAddress com um endereço IP que não esteja atribuído a um *host*?

11-14 Escreva um código em Java para criar um endereço de *socket* associado ao *host* local e ao processo-servidor do tipo HTTP.

11-15 Escreva um código em Java para criar um endereço de *socket* associado ao *host* local e ao número de porta efêmera 56000.

11-16 Escreva um código em Java para criar um endereço de *socket* associado ao *host* com o nome de domínio "algum.com" e a um processo-cliente com o número de porta 51000.

11-17 Escreva um código em Java para criar um endereço de *socket* associado ao InetSocketAddress `addr` e ao processo-servidor do tipo TELNET.

11-18 Escreva um código em Java para recuperar o InetAddress de um InetSocketAddress nomeado sockAd.

11-19 Escreva um código em Java para recuperar o número de porta de um InetSocketAddress nomeado sockAd.

11-20 Qual é a diferença entre a classe DatagramSocket e a classe Socket em Java?

11-21 Qual é a diferença entre a classe Server Socket e a classe Socket em Java?

11-22 Dizemos que, na programação de redes, um *socket* deve ao menos estar associado a um endereço local de *socket*. O primeiro construtor da classe DatagramSocket (ver Tabela 11.6) não tem parâmetros. Você pode explicar como ele é associado ao endereço de *socket* local quando é usado no lado do cliente?

11-23 A classe DatagramPacket tem dois construtores (ver Tabela 11.7). Qual construtor pode ser usado para criar um pacote de envio?

11-24 A classe DatagramPacket tem dois construtores (ver Tabela 11.7). Qual construtor pode ser usado para criar um pacote de recepção?

11-25 Na Figura 11.2, quais são as alterações necessárias se um cliente precisar enviar uma mensagem diferente de uma cadeia de

caracteres (*string*), como uma imagem, por exemplo?

11-26 Na Figura 11.3, considere que o pedido consiste em um URL para recuperar uma imagem. Como o URL é armazenado na variável recvBuff?

11-27 Explique como um programa cliente UDP (Tabela 11.8) aguarda até que a resposta venha do servidor.

11-28 Como um objeto da classe Socket em um cliente TCP (Figura 11.5) é criado e destruído?

11-29 Existem classes de fluxo de entrada em Java. Você pode explicar por que o programa-cliente TCP não usa diretamente essas classes para criar o fluxo de entrada?

11-30 Na Figura 11.5, como o cliente cria os fluxos de entrada e de saída para se comunicar com o servidor?

Problemas

11-1 Escreva um método em Java que aceite uma cadeia de caracteres (*string*) representando um endereço IP na forma "*x.y.z.t*" e a transforme em um inteiro sem sinal.

11-2 Escreva um método em Java que converta um número inteiro de 32 *bits* em uma cadeia de caracteres (*string*) representando um endereço IP na forma "*x.y.z.t*".

11-3 Escreva um método em Java que extraia o prefixo de um endereço (na forma de um número inteiro) dada uma cadeia de caracteres (*string*) na notação CIDR, na forma "*x.y.z.t/n*".

11-4 Escreva um método em Java que recupere, a partir de uma cadeia de caracteres na notação CIDR ("*x.y.z.t/n*"), o endereço IP (sem prefixo) na forma de uma cadeia de caracteres (*string*) em notação decimal com pontos.

11-5 Escreva um método em Java que adicione um dado prefixo (na forma de um número inteiro) ao final de um endereço IP, criando uma cadeia de caracteres (*string*) na notação CIDR ("*x.y.z.t/n*").

11-6 Escreva um método em Java que transforme um número inteiro representando um prefixo em um número inteiro de 32 *bits* sem sinal representando a máscara numérica correspondente.

11-7 Escreva um método em Java que transforme um número inteiro sem sinal de 32 *bits* representando uma máscara em um número inteiro representando um prefixo (/n).

11-8 Escreva um método em Java que determine o primeiro endereço (endereço de rede) de um bloco quando um dos endereços do bloco é dado na forma de uma cadeia de caracteres (*string*) na notação CIDR.

11-9 Escreva um método em Java que determine o último endereço em um bloco quando um dos endereços do bloco é fornecido na forma de uma cadeia de caracteres (*string*) na notação CIDR.

11-10 Escreva um método em Java que determine o tamanho de um bloco dado um endereço nesse bloco (em notação CIDR).

11-11 Escreva um método em Java que determine o tamanho da faixa de endereços quando os endereços de início e de fim são fornecidos.

11-12 Refaça o Exemplo 11.4 usando a abordagem concorrente.

11-13 Refaça o Exemplo 11.4 usando o serviço do TCP.

11-14 Refaça o Problema 11.13 usando a abordagem concorrente.

11.6 TAREFAS DE PROGRAMAÇÃO

Prg11-1 Modifique, compile e teste o programa-cliente da Tabela 11.8 e o programa-servidor da Tabela 11.9 de modo que eles façam o seguinte: o programa-cliente deve ler a cadeia de caracteres (*string*) do pedido a partir de um arquivo e salvar a cadeia de caracteres de resposta em outro arquivo. O nome do arquivo deve ser passado como

argumento para o método main do programa-cliente. O programa-servidor deve aceitar a cadeia de caracteres que representa o pedido, transformar todas as letras minúsculas em maiúsculas e retornar o resultado.

Prg11-2 Modifique, compile e teste o programa-cliente da Tabela 11.13 e o programa-servidor da Tabela 11.14 de modo a permitir que o cliente forneça o caminho de um pequeno arquivo armazenado no servidor. O servidor deve enviar o conteúdo do arquivo na forma de uma cadeia de caracteres. O cliente armazena o arquivo na máquina cliente. Esse exemplo permite simular um protocolo simples de transferência de arquivos.

Prg11-3 Modifique, compile e teste o programa-cliente da Tabela 11.13 e o programa-servidor da Tabela 11.14 de modo a simular um cliente e um servidor DNS locais. O servidor possui uma pequena tabela que contém duas colunas, o nome do domínio e o endereço IP. O cliente pode enviar dois tipos de pedidos: normal e reverso. O pedido normal é uma cadeia de caracteres no formato "*N:nome de domínio*"; o pedido reverso é uma cadeia de caracteres no formato "*R:endereço IP*". O servidor responde com o endereço IP, o nome de domínio ou a mensagem "Não encontrado".

Prg11-4 Escreva programas cliente-servidor concorrentes baseados em TCP para simular uma versão simplificada do HTTP (Capítulo 2) usando apenas uma conexão não persistente. O cliente envia uma mensagem HTTP; o servidor responde com o arquivo solicitado. Use somente dois tipos de métodos, GET e PUT, e apenas alguns cabeçalhos simples. Perceba que, após testar o programa resultante, você deve ser capaz de testar o mesmo programa com um navegador Web.

Prg11-5 Escreva programas cliente-servidor concorrentes baseados em TCP para simular uma versão simplificada do POP (Capítulo 2). O cliente envia um pedido para receber um e-mail em sua caixa de correio; o servidor responde com o e-mail.

GLOSSÁRIO

acesso aleatório Uma categoria de acesso ao meio em que cada estação pode acessar o meio sem ser controlada por qualquer outra estação.

acesso controlado Um método de acesso múltiplo no qual as estações consultam umas às outras para determinar quem tem o direito de enviar dados.

acesso local Ato de acessar uma estação usando o terminal diretamente conectado àquela estação.

acesso múltiplo (MA – multiple access) Um método de acesso ao meio em que cada estação pode acessar o meio livremente.

Acesso Múltiplo com Verificação de Portadora (CSMA – Carrier Sense Multiple Access) Um método de acesso baseado em contenção no qual cada estação verifica o meio antes de transmitir dados.

Acesso Múltiplo com Verificação de Portadora / Detecção de Colisão (CSMA/CD – Carrier Sense Multiple Access with Collision Detection) Um método de acesso no qual as estações transmitem sempre que o meio de transmissão fica disponível e retransmitem os dados quando ocorre colisão.

Acesso Múltiplo com Verificação de Portadora / Prevenção de Colisão (CSMA/CA – Carrier Sense Multiple Access with Collision Avoidance) Um método de acesso em redes sem fio que busca evitar colisão ao obrigar que as estações enviem mensagens de reserva ao perceber que o canal está ocioso.

Acesso Múltiplo por Divisão de Código (CDMA – Code Division Multiple Access) Um método de acesso múltiplo no qual um canal transporta todas as transmissões simultaneamente.

Acesso Múltiplo por Divisão de Frequência (FDMA – Frequency-Division Multiple Access) Uma método de acesso no qual cada uma de múltiplas fontes usa uma largura de banda que lhe é atribuída em uma banda de comunicação de dados.

Acesso Múltiplo por Divisão de Tempo (TDMA – Time-Division Multiple Access) Um método de acesso múltiplo no qual a largura de banda consiste em apenas um canal de tempo compartilhado.

acesso remoto O processo de acessar um computador remoto a partir de um terminal conectado a um computador local.

Agência de Projetos de Pesquisa Avançados (ARPA – Advanced Research Projects Agency) A agência do governo que criou a ARPANET. Mais tarde, a mesma agência criou a Internet global.

Agência de Projetos de Pesquisa Avançados de Defesa (DARPA – Defense Advanced Research Projects Agency) Uma organização governamental que, sob o nome de ARPA, fundou a ARPANET e a Internet.

Agência de Segurança Nacional (NSA – National Security Agency) Um agência de inteligência dos Estados Unidos que trabalha com assuntos de segurança.

Agente de Acesso a Mensagens (MAA – Message Access Agent) Um programa cliente-servidor que recebe os e-mails armazenados em um servidor.

agente de retransmissão Para o BOOTP, um roteador que pode ajudar a enviar pedidos locais para servidores remotos.

Agente de Transferência de Mensagens (MTA – Mail Transfer Agent) Um componente do SMTP que transfere mensagens de e-mail na Internet.

Agente de Transferência de Mensagens (MTA – Message Transfer Agent) Um componente do SMTP que transmite a mensagem via Internet.

Agente de Usuário (UA – User Agent) Um componente do SMTP que prepara a mensagem, cria o envelope e coloca a mensagem no envelope.

agente externo (*foreign agent*) No contexto de IP móvel, o agente externo é um roteador ou um *host* conectado à rede externa. O agente externo recebe e entrega pacotes enviados pelo agente local para o nó móvel.

agente nativo (*home agent*) Normalmente, um roteador conectado à rede doméstica do nó móvel que recebe e envia pacotes (tendo o nó móvel como destino) pelo agente externo (*foreign agent*).

agregação de endereços Um mecanismo no qual os blocos de endereços de várias organizações são agregados em um bloco maior.

alerta (*trap*) No SNMP, uma PDU enviada por um agente para o gerente para relatar um evento.

algoritmo de cifração Um algoritmo que converte uma mensagem em uma forma ininteligível, a qual não pode ser compreendida a menos que seja decifrada.

algoritmo de decifração Algoritmo para desembaralhar o texto cifrado, levando ao texto às claras original.

algoritmo de Dijkstra Um algoritmo usado em roteamento baseado no estado de enlace para encontrar o caminho mais curto entre roteadores.

Algoritmo de *Hash* Seguro (SHA – Secure Hash Algorithm) Uma série de padrões de função de *hash* promovidos pelo NIST e publicado em documentações conhecidas como FIPS. Os algoritmos originais se baseiam fortemente em algoritmos da família MD, como o MD5.

Algoritmo de Karn Um algoritmo que não leva em consideração os segmentos retransmitidos no cálculo do RTT.

algoritmo de Nagle Um algoritmo que busca prevenir a síndrome da janela boba (*silly window syndrome*) no lado do emissor; as taxas de geração de dados e a velocidade da rede são ambas levadas em consideração.

algoritmo de prevenção de congestionamento (*congestion avoidance*) Um algoritmo usado pelo TCP que busca evitar o congestionamento através da redução da velocidade de transmissão.

algoritmo do balde furado (*leaky bucket*) Um algoritmo para controlar tráfego em rajadas.

ALOHA O método original de acesso múltiplo e aleatório no qual uma estação pode enviar um quadro a qualquer momento em que ela tenha algo para enviar.

ALOHA puro O ALOHA original, que não usa parcelas discretas (*slots*) de tempo.

ALOHA segmentado (*slotted ALOHA*) Uma variante do método de acesso ALOHA em que o tempo é dividido em parcelas de tempo discretas (*slots*) e cada estação é obrigada a iniciar o envio de dados apenas no início de uma parcela.

amostra e retém (*sample and hold*) Um método de amostragem que faz a amostragem da amplitude de um sinal analógico e mantém o valor até a amostra seguinte.

amostragem O processo de obtenção de amplitudes de um sinal em intervalos regulares.

amplitude A força de um sinal, geralmente medida em volts.

amplitude de pico A amplitude máxima de um sinal analógico.

AMPS digital (D-AMPS – DigitalAMPS) Um sistema de telefonia celular da segunda geração, que consiste em uma versão digital do AMPS.

análise de Fourier A técnica matemática utilizada para obter o espectro de frequências de um sinal aperiódico se sua representação no domínio do tempo for fornecida.

ângulo crucial Em óptica, refere-se ao ângulo de incidência de um raio que leva à mudança do seu comportamento de refração para reflexão.

ângulo de incidência Em óptica, refere-se ao ângulo formado entre um raio de luz que se aproxima da interface entre dois meios e a linha perpendicular à interface.

Antena de Múltiplas Entradas e Múltiplas Saídas (MIMO – Multiple-Input and Multiple-Output) Um conjunto de antenas inteligentes proposto para sistemas 4G sem fio que permite a transmissão de fluxos independentes simultaneamente a partir de todas as antenas para aumentar a taxa de dados múltiplas vezes.

antena de TV comunitária (CATV – community antenna TV) Um serviço de rede a cabo que transmite sinais de vídeo em locais onde há baixa ou nenhuma recepção.

antena omnidirecional Uma antena que envia ou recebe sinais em todas as direções.

antena tipo corneta Uma antena em forma de concha usada na comunicação por micro-ondas terrestres.

antena unidirecional Uma antena que envia ou recebe sinais em uma única direção.

Anúncio de Estado de Enlace (LSA – Link-State Advertisement) Um método para disseminar informações usado no OSPF.

applet Um programa de computador para a criação de um documento ativo na Web. Geralmente, é escrito em Java.

apresentação em três vias (*three-way handshake*) Um sequência de eventos para o estabelecimento ou encerramento de conexão que consiste na solicitação, seguida da confirmação da solicitação e finalmente no reconhecimento da confirmação.

área de cobertura Uma área da Terra coberta por um satélite em um certo momento.

área Uma coleção de redes, estações e roteadores dentro de um sistema autônomo.

aritmética modular Aritmética que usa uma faixa limitada de inteiros (0 até $n = 1$).

armazenamento em *cache* O armazenamento de informações em uma memória rápida e pequena.

Árvore Baseada em Núcleo (CBT – Core-Based Tree) Um protocolo *multicast* de árvore compartilhada que usa um roteador central como raiz da árvore.

árvore baseada na origem Uma árvore utilizada para *multicast* usada por protocolos de *multicast* nos quais uma única árvore é criada para cada combinação de origem e grupo.

árvore compartilhada Uma característica do roteamento *multicast* em que cada grupo no sistema compartilha a mesma árvore.

árvore de abrangência (*spanning tree*) Uma árvore tendo a origem como raiz e os membros do grupo como folhas; uma árvore que conecta todos os nós.

árvore de custo mínimo No contexto de roteamento de menor custo, refere-se a uma árvore que cobre todo o gráfico da rede, tendo a origem como sua raiz.

árvore de menor custo Uma tabela de roteamento formada usando o algoritmo de Dijkstra.

árvore de pontos de encontro Um método de árvore de grupo compartilhada no qual há uma árvore para cada grupo.

assinatura digital Um mecanismo de segurança no qual o remetente pode assinar eletronicamente uma mensagem e o destinatário pode verificar a mensagem e provar que ela foi realmente assinado pelo remetente.

Associação de Segurança (SA – Security Association) Um protocolo IPSec que cria uma conexão lógica entre dois *hosts*.

associação Uma conexão no SCTP.

ataque de inundação de SYN Um problema sério de segurança na fase de estabelecimento de conexão do TCP no qual um ou mais atacantes maliciosos enviam um grande número de segmentos de SYN.

ataque de repetição O reenvio de uma mensagem que foi interceptada por um intruso.

ataque *man-in-the-middle* (homem no meio) Um ataque no qual um intruso intercepta e envia mensagens entre o emissor e o receptor verdadeiros durante um protocolo de estabelecimento de chaves.

atenuação A perda de energia de um sinal devido à resistência do meio.

atraso de propagação *Ver* tempo de propagação.

áudio Gravação ou transmissão de som ou música.

Áudio MPEG Camada 3 (MP3 – MPEG audio layer 3) Um padrão que usa codificação perceptual para comprimir áudio.

áudio/vídeo interativo Comunicação em tempo real com som e imagem.

aumento aditivo No TCP, é uma estratégia de controle de congestionamento na qual o tamanho da janela é aumentado de um único segmento em vez de exponencialmente.

Aumento Aditivo, Redução Multiplicativa (AIMD – Additive Increase, Multiplicative Decrease) Combinação dos métodos de controle de congestionamento "aumento aditivo" e "redução multiplicativa" utilizados no TCP.

autenticação de entidade Uma técnica projetada para permitir que uma entidade comprove a identidade de outra entidade.

autenticação de mensagem Uma medida de segurança na qual o remetente da mensagem é verificado em cada mensagem enviada.

autenticação de usuário Uma medida segurança na qual a identidade do remetente é verificada antes do início de uma comunicação.

autenticação por desafio-resposta Um método de autenticação em que o requerente prova que sabe um segredo sem enviá-lo.

autenticação Verificação do remetente de uma mensagem.

autonegociação Uma característica do Fast Ethernet que permite que dois dispositivos negociem o modo ou a taxa de transmissão de dados.

Autoridade Certificadora (AC) Uma agência, como uma organização federal ou estadual, que associa uma chave pública a uma entidade e lhe emite um certificado.

Autoridade para Atribuição de Números na Internet (IANA – Internet Assigned Numbers Authority) Um grupo apoiado pelo governo dos Estados Unidos que era responsável pela gestão dos nomes de domínio e endereços na Internet até outubro de 1998.

autossincronização Sincronização de longas sequências de 1s ou 0s por meio do método de codificação.

balde de fichas Um algoritmo que permite que *hosts* ociosos acumulem crédito, em forma de fichas, para transmissões futuras.

banco de dados distribuído Informação armazenada em diversos locais.

banda de guarda A largura de banda que separa dois sinais.

Base de Dados de Associações de Segurança (SAD – Security Association Database) Uma tabela bidimensional na qual cada linha define uma única associação de segurança (SA – Security Association).

Base de Dados de Estado de Enlace (LSDB – Link State Database) No contexto de roteamento baseado no estado de enlace (*link-state*), refere-se a um banco de dados comum a todos os roteadores e preenchido a partir de informações de pacotes de estado de enlace.

Base de Dados de Políticas de Segurança (SPD – Security Policy Database) Uma base de dados de políticas de segurança (SPs).

Base de Informações de Gerenciamento (MIB – Management Information Base) A base de dados usada pelo SNMP, a qual contém as informações necessárias para o gerenciamento de uma rede.

baud O número de elementos de sinal transmitidos por segundo. Um elemento de sinal é constituído por um ou mais *bits*.

Bellman-Ford Um algoritmo usado para calcular tabelas de roteamento no método de roteamento por vetor de distâncias.

bilhete (*ticket*) Uma mensagem cifrada destinada à entidade B, mas enviada à entidade A para que seja entregue.

Bipolar com Substituição de 8 Zeros (B8ZS – Bipolar with 8-Zero Substitution) Uma técnica de embaralhamento na qual uma sequência de oito zeros é substituída por uma sequência predefinida para melhorar a sincronização de *bits*.

Bipolar de Alta Densidade de 3 Zeros (HDB3 – High-Density Bipolar 3-Zero) Uma técnica de embaralhamento na qual quatro voltagens no nível zero consecutivas são substituídas por uma de duas sequências predefinidas.

***bit* de início** Na transmissão assíncrona, um *bit* para indicar o início da transmissão.

***bit* de parada** Na transmissão assíncrona, um ou mais *bits* para indicar o fim da transmissão.

***bit* de varredura/final (P/F – poll/final)** Um *bit* no campo de controle do HDLC; se a estação primária estiver enviando dados, o *bit* assume o valor de varredura; se a estação secundária estiver enviando dados, ele assume o valor de final.

bit Dígito binário. A menor unidade de dados possível (0 ou 1).

bloco (pedaço ou *chunk*) Uma unidade de transmissão no SCTP.

Bluetooth Uma tecnologia de LAN sem fio projetada para conectar dispositivos de diferentes tipos, como telefones e *notebooks*, em uma área pequena, como uma sala.

***Broadcast* pelo Caminho Reverso (RPB – Reverse Path Broadcasting)** No contexto de transmissão *multicast*, refere-se a uma técnica em que é garantido que cada destino recebe uma e apenas uma cópia do pacote.

***bucket brigade attack* (ataque brigada de incêndio)** *Ver* ataque *man-in-the middle*.

***buffer* de reprodução** Um *buffer* que armazena os dados antes que eles estejam prontos para serem reproduzidos.

buffer Memória reservada para armazenamento temporário.

byte Um conjunto de 8 *bits*. Um octeto.

cabeçalho (*header*) informação de controle adicionada ao início de um pacote de dados.

cabo coaxial Um meio de transmissão que consiste em um núcleo condutor, material isolante e um segundo revestimento condutor.

cabo de fibra óptica Um meio de transmissão de banda larga que transporta sinais de dados na forma de pulsos de luz. É constituído por um cilindro fino de vidro ou de plástico, chamado núcleo, recoberto por uma camada concêntrica de vidro ou de plástico conhecido como revestimento ou *cladding*.

cabo de par trançado Um meio de transmissão que consiste em dois condutores isolados em uma configuração torcida.

caixa P (*P-Box*) Um componente das cifras de bloco modernas que faz uma transposição dos *bits*.

caixa S (*S-Box*) Um elemento usado em criptografia composto por decodificadores, caixas P e codificadores.

cálculo de *hash* Uma técnica em que um resumo criptográfico de tamanho fixo é criado a partir de uma mensagem de comprimento arbitrário.

camada ATM Uma camada do ATM que provê serviços de roteamento, gerenciamento de tráfego, comutação e multiplexação.

Camada de Adaptação ATM (AAL – ATM Adaptation Layer) A camada do protocolo ATM que encapsula os dados do usuário.

Camada de Adaptação Simples e Eficiente (SEAL – Simple and Efficient Adaptation Layer) Uma camada AAL projetada para a Internet (AAL5).

camada de aplicação A quinta camada do modelo Internet; fornece acesso aos recursos da rede.

camada de apresentação A sexta camada do modelo OSI; responsável pela tradução, cifração, autenticação e compressão de dados.

camada de enlace A segunda camada do modelo Internet. É responsável pela comunicação nó a nó.

camada de rede A terceira camada no modelo de Internet, responsável pela entrega de um pacote para o destino final.

camada de sessão A quinta camada do modelo OSI, responsável pelo estabelecimento, gerenciamento e finalização de conexões lógicas entre dois usuários finais.

camada de transporte A terceira camada no modelo Internet e a quarta do modelo OSI; responsável pela entrega fim a fim confiável e pela recuperação de erros.

camada física A primeira camada do modelo Internet, responsável pelas especificações mecânicas e elétricas do meio.

Caminho de Transmissão (TP – Transmission Path) No ATM, uma conexão física entre um ponto final e um *switch* ou entre dois *switches*.

Caminho Virtual (VP – Virtual Path) Uma combinação de circuitos virtuais no ATM.

canal com ruído Um canal que pode introduzir erros nos dados sendo transmitidos.

canal passa-baixas Um canal que permite a passagem de frequências entre 0 e *f*.

canal passa-faixa Um canal que permite a passagem de uma faixa de frequências.

canal sem ruído Um canal livre de erros.

canal Síncrono Orientado a Conexão (SCO – Synchronous Connection-Oriented) Em uma rede Bluetooth, corresponde a uma ligação física criada entre um mestre e um escravo que reserva parcelas de tempo (*slots*) específicas em intervalos regulares.

canal Uma via de comunicação.

canalização Um método de acesso múltiplo em que a banda disponível em um enlace é compartilhada no tempo.

capacidade de Shannon A taxa de dados mais alta, teoricamente, para um canal.

caractere de escape Um caractere usado para alterar o significado do caractere seguinte.

carimbo de tempo (*timestamp*) Um campo no pacote relacionado com o tempo absoluto ou relativo em que o pacote é criado ou enviado.

CDMA Multiportadora (MC-CDMA – Multi-Carrier CDMA) Um método de acesso proposto para redes sem fio 4G.

célula Uma unidade de dados de tamanho pequeno e fixo; além disto, na telefonia celular, refere-se a uma área geográfica servida por uma estação-base.

Central de Comutação de Telefonia Móvel (MTSO – mobile telephone switching office) Uma estação que controla e coordena a comunicação entre todas as estações celulares e a estação de controle de telefonia.

Central de Comutação Móvel (MSC – Mobile Switching Center) No contexto de telefonia celular, uma estação de comutação que coordena a comunicação entre todas as estações-base e a estação central de telefonia.

central de comutação O local em que as centrais telefônicas ficam localizadas.

central local (*end office*) Uma central de comutação que serve como terminal para os lacetes locais.

Centro de Distribuição de Chaves (KDC – Key-Distribution Center) No contexto de criptografia de chave secreta, refere-se a uma entidade confiável (TTP – Trusted Third Party), que compartilha uma chave com cada usuário.

certificado de chave pública Um certificado que define o proprietário de uma chave pública.

***Challenge Handshake Authentication Protocol* (CHAP)** No PPP, é um protocolo de apresentação em três vias (*three-way handshaking*) utilizado para autenticação.

chave privada Em um sistema de cifração baseado em chave assimétrica, refere-se à chave usada para decifrar. Em uma assinatura digital, refere-se à chave usada para assinar.

chave pública Em um sistema de cifração baseado em chave assimétrica, refere-se à chave usada para cifrar. Em uma assinatura digital, refere-se à chave utilizada para verificar a assinatura.

chave Um conjunto de valores usado por uma cifra durante sua operação.

Chaveamento de Código Complementar (CCK – Complementary Code Keying) Um método de codificação HR-DSSS que codifica quatro ou oito *bits* em um símbolo.

Chord Um protocolo P2P proposto por Stoica *et al.* em 2001, no qual o espaço de identificadores é decomposto em 2^m pontos distribuídos em um círculo em sentido horário.

cifra (*cipher*) Um algoritmo de cifração e/ou decifração.

cifra aditiva A cifra monoalfabética mais simples de todas, na qual cada caractere é cifrado pela adição de seu valor ao da chave.

cifra de autochave (*autokey cipher*) Uma cifra de fluxo na qual cada subchave do fluxo é a mesma que o caractere anterior do texto às claras. A primeira subchave é o segredo compartilhado entre as partes.

cifra de bloco Um tipo de cifra em que blocos de texto às claras são cifrados um de cada vez, usando a mesma chave secreta.

cifra de César Uma cifra de permutação utilizada por Júlio César com o valor da chave fixado em 3.

cifra de chave assimétrica Uma cifra que emprega um sistema criptográfico de chave assimétrica.

cifra de chave simétrica Uma cifra usando um sistema de criptografia de chave simétrica.

cifra de deslocamento Um tipo de cifra aditiva na qual a chave define o deslocamento de caracteres na direção do final do alfabeto.

cifra de Feistel Uma classe de cifras que consistem tanto em componentes inversíveis como não inversíveis.

cifra de fluxo Um tipo de cifra na qual a cifração e decifração são feitas um símbolo (como um caractere ou um *bit*) de cada vez.

cifra de substituição Uma cifra que substitui um símbolo por outro.

cifra de transposição Uma cifra que transpõe símbolos no texto às claras para criar o texto cifrado.

cifra de uso único (*one-time pad*) Uma cifra inventada por Vernam na qual a chave é uma sequência aleatória de símbolos que têm o mesmo comprimento que o texto às claras.

cifra de Vigenère Um esquema de substituição polialfabética que usa a posição de um caractere no texto às claras e a posição do caractere no alfabeto.

cifra monoalfabética Uma cifra de substituição na qual um símbolo no texto original é sempre transformado no mesmo símbolo na mensagem cifrada, independentemente da sua posição no texto.

cifra polialfabética Uma cifra na qual cada ocorrência de um caractere pode ter um substituto diferente.

cifração Processo de conversão de uma mensagem em uma forma ininteligível, a qual não pode ser compreendida a menos que seja decifrada.

Circuito Virtual (VC – Virtual Circuit) Um circuito lógico formado entre os computadores emissor e receptor. **comutação de circuitos virtuais** Um técnica de comutação usada em WANs comutadas.

Circuito Virtual Comutado (SVC – Switched Virtual Circuit) Um método de transmissão baseado em circuitos virtuais no qual um circuito virtual é criado e existe apenas durante o período da transmissão.

Circuito Virtual Permanente (PVC – Permanent Virtual Circuit) Um método de transmissão de circuito virtual no qual o mesmo circuito virtual é usado entre origem e destino em uma base contínua.

cliente Um programa que pode utilizar serviços de programas chamados servidores.

Codificação 2-Binário, 1-Quaternário (2B1Q) Uma técnica de codificação de linha em que cada pulso representa 2 *bits*.

codificação 4B/5B (4-binário, 5-binário) Uma técnica de codificação de bloco na qual um bloco de 4 *bits* é codificado em um código de 5 *bits*.

codificação 8B/10B (8-binário, 10-binário) Uma técnica de codificação de bloco na qual 8 *bits* são codificados em um código de 10 *bits*.

codificação 8B6T (8-binário, 6-ternário) Um esquema de codificação de linha de três níveis, o qual codifica um bloco de 8 *bits* em um sinal de seis pulsos ternários.

codificação aritmética Uma codificação sem perdas na qual a mensagem toda é mapeada para um pequeno intervalo dentro do intervalo [0 ; 1). O intervalo pequeno é então codificado como uma sequência binária.

codificação bipolar Um método de codificação digital-digital no qual a amplitude 0 representa o valor 0 enquanto amplitudes positivas e negativas representam 1s alternados.

codificação de bloco Um método de codificação no qual blocos de n *bits* são codificados em blocos de m *bits*, onde $m > n$.

codificação de linha Ato de converter dados binários em sinais.

codificação do tipo transmissão por múltiplas linhas com 3 níveis (MLT-3 – multiline transmission, 3-level) Um esquema de codificação em linha que apresenta três níveis de sinais e transições no início do *bit* 1.

codificação Manchester diferencial Um método de codificação polar digital-digital que apresenta uma transição no meio do intervalo de *bit*, bem como uma inversão no início de cada *bit* 1.

codificação Manchester Um método de codificação polar digital-digital em que uma transição ocorre no meio de cada intervalo de *bit* para garantir a sincronização.

codificação perceptual A técnica de compressão de modo geral usada para gerar áudio com qualidade de CD, com base na ciência da psicoacústica. Os algoritmos utilizados na codificação perceptual primeiro transformam os dados do domínio de tempo para o domínio de frequência; as operações são, então, realizadas sobre os dados no domínio da frequência.

codificação polar Um método de codificação digital-analógico que utiliza dois níveis de amplitude (positivo e negativo).

Codificação por Carreira (RLE – Run-Length Coding) Um método de compressão para remoção de redundâncias. O método substitui uma sequência repetida do mesmo símbolo por dois elementos: um valor de contagem e o símbolo em si.

Codificação Preditiva (PC – Predictive Coding) No contexto de compressão de áudio, refere-se à codificação apenas das diferenças entre as amostras.

Codificação Preditiva Linear (LPC – Linear Predictive Coding) Um método de codificação preditiva no qual, em vez de enviar sinais de diferenças quantizadas, o emissor analisa os sinais e determina as suas características.

codificação unipolar Um método de codificação digital-digital em que um valor diferente de zero representa 1 ou 0; o outro *bit* é representado pelo valor zero.

código cíclico Um código linear no qual o deslocamento cíclico (rotação) de cada palavra do código cria uma outra palavra do código.

Código de Autenticação de Mensagens (MAC – Message Authentication Code) Um resumo criptográgico cujo cálculo envolve um segredo entre duas partes.

código de bloco linear Um código de bloco em que a adição de duas palavras de código cria uma outra palavra de código.

Código de Detecção de Modificação (MDC – Modification Detection Code) O resumo criado por uma função de *hash*.

código de Hamming Um método que adiciona *bits* redundantes em uma unidade de dados para detectar e corrigir erros.

código de Huffman Um método de compressão estatística que usa códigos de comprimento variável para codificar um conjunto de símbolos.

código de verificação de paridade Um método de detecção de erros usando um *bit* de paridade.

Código Estendido de Intercâmbio Decimal Codificado em Binário (EBCDIC – Extended Binary Coded Decimal Interchange Code) Um código de caracteres de 8 *bits* desenvolvido e usado pela IBM.

Código Padrão Americano para Intercâmbio de Informações (ASCII – American Standard Code for Information Interchange) Uma codificação de caracteres desenvolvida pela ANSI e amplamente utilizada na comunicação de dados.

ColdFusion Uma tecnologia Web dinâmica que permite a fusão de dados provenientes de um banco de dados convencional.

colisão forte Criação de duas mensagens com o mesmo resumo criptográfico.

colisão ● evento que ocorre quando dois emissores enviam dados ao mesmo tempo em um canal de transmissão concebido para comportar apenas uma transmissão de cada vez; os dados são destruídos.

Comissão Federal de Comunicações (FCC – Federal Communications Commission) Uma agência governamental americana que regula as transmissões de rádio e televisão e as telecomunicações.

Comitê Consultivo Internacional de Telegrafia e Telefonia (CCITT) Um grupo de normalização internacional atualmente conhecido como ITU-T.

complemento de um Uma representação de números binários na qual o complemento de um número é determinado pela complementação de todos os seus *bits*.

comportamento por salto (PHB – per-hop behavior) No modelo Diffserv, um campo de 6 *bits* que define o mecanismo de tratamento de pacotes para cada pacote.

compressão com perdas Um método de compressão em que a perda de alguns dados é realizada para se obter uma melhor taxa de compressão.

compressão espacial Ato de comprimir uma imagem por meio da remoção de redundâncias.

compressão sem perdas Um método de compressão em que a integridade dos dados é preservada dado que os algoritmos de compressão e descompressão são inversos exatos um do outro: nenhuma parte dos dados é perdida no processo.

compressão temporal Um método de compressão MPEG no qual os quadros redundantes são removidos.

comprimento de *bit* (*bit length*) ● comprimento de um *bit* sendo transferido, medido em metros.

comprimento de onda A distância que um sinal simples pode viajar em um período.

Comunicação para Internet Móvel (IMT-2000 – Internet Mobile Communication 2000) Um modelo criado pela ITU que define critérios para a terceira geração da telefonia celular.

comunicação processo a processo A comunicação entre dois aplicativos em execução.

comutação de circuitos Uma tecnologia de comutação que estabelece uma conexão elétrica entre as estações utilizando um caminho dedicado.

comutação de pacotes A transmissão de dados usando uma rede de comutação de pacotes.

comutação por divisão de espaço Comutação na qual os caminhos são separados uns dos outros espacialmente.

comutação por divisão de tempo Uma técnica de comutação de circuitos na qual a multiplexação por divisão de tempo é usada para se fazer a comutação.

comutador *banyan* Um comutador multiestágio contendo microcomutadores em cada estágio, os quais são responsáveis por encaminhar os pacotes de acordo com a porta de saída, representada como uma sequência binária.

comutador Batcher-banyan Um comutador *banyan* que ordena os pacotes entrantes com base na sua porta de destino.

comutador *crossbar* Um comutador que consiste em um reticulado de caminhos horizontais e verticais. Na intersecção de cada caminho horizontal e vertical, existe um ponto de cruzamento (*crosspoint*) que pode conectar a entrada à saída.

comutador multiestágio Um tipo de comutador projetado para reduzir o número de pontos de cruzamento (*crosspoints*).

conector Bayone-Neill-Concelman (BNC) Um conector de cabo coaxial comum.

conector de ponta reta Um tipo de conector de cabo de fibra óptica que usa um sistema de fecho tipo baioneta.

conexão não persistente Um tipo de conexão no qual uma conexão TCP é feita para cada pedido/resposta.

conexão persistente Uma conexão em que o servidor deixa a conexão aberta para receber mais pedidos após o envio de uma resposta.

conexão ponto a ponto Um enlace de transmissão dedicado entre dois dispositivos.

confiabilidade Uma característica de QoS de um fluxo; indica o quanto se pode confiar na transmissão. Uma rede é confiável quando não corromper, perder ou duplicar um pacote.

configuração desbalanceada Uma configuração HDLC na qual um dispositivo é primário e os outros são secundários.

confirmação (ACK – acknowledgment) Uma resposta enviada pelo receptor para indicar a recepção bem-sucedida de dados.

conformação de tráfego (*traffic shaping*) Um mecanismo para melhorar a QoS que controla a quantidade e a taxa do tráfego enviado para a rede.

congestionamento Tráfego de rede ou entre redes excessivo, causando uma degradação geral do serviço.

Conjunto Básico de Serviços (BSS – Basic Service Set) O bloco construtivo de uma LAN sem fio, conforme definido pelo padrão IEEE 802.11.

Conjunto de algoritmos criptográficos (*cipher suite*) Uma lista de cifras possíveis.

Conjunto Extendido de Serviços (ESS – Extended Service Set) Um serviço de LANs sem fio composto por dois ou mais BSS com pontos de acesso, conforme definido pelo padrão IEEE 802.11.

Conselho de Arquitetura da Internet (IAB – Internet Architecture Board) Órgão de consultoria técnica da ISOC; supervisiona o desenvolvimento contínuo da pilha de protocolos TCP/IP.

contagem de saltos (*hop count*) O número de nós ao longo de uma rota. É uma medida de distância usada em algoritmos de roteamento.

contenção Um método de acesso no qual dois ou mais dispositivos tentam transmitir ao mesmo tempo no mesmo canal.

controle de congestionamento em malha aberta (*open-loop*) Aplicação de conjunto de políticas para prevenir o congestionamento.

controle de congestionamento em malha fechada (*closed-loop*) Um método para minimizar o congestionamento depois que ele ocorre.

controle de congestionamento O mecanismo usado para eliminar ou evitar congestionamento na rede.

Controle de Enlace de Dados (DLC – Data-Link Control) As responsabilidades da camada de enlace de dados: controle de fluxo e controle de erros.

Controle de Enlace de Dados de Alto Nível (HDLC – High-level Data Link Control) Um protocolo da camada de enlace orientado a *bits* que foi definido pela ISO.

Controle de Enlace Lógico (LLC – Logical Link Control) A subcamada superior da camada de enlace de dados tal como definido pelo Projeto IEEE 802.2.

controle de erros O tratamento de erros na transmissão de dados.

controle de fluxo Uma técnica para controlar a taxa de fluxo de quadros (pacotes ou mensagens).

controle de tráfego Um método para moldar e controlar o tráfego em uma rede de longa distância.

convergência lenta Uma deficiência do RIP que aparece quando uma mudança em algum lugar na Internet se propaga muito lentamente pelo restante da Internet.

conversão analógico-analógico A representação de informação analógica por meio de um sinal analógico.

conversão analógico-digital A representação de informação analógica por meio de um sinal digital.

conversão digital-analógico A representação de informação digital usando um sinal analógico.

conversão digital-digital A representação de dados digitais por um sinal digital.

***cookie* mágico (*magic cookie*)** No DHCP, refere-se ao número no formato de um endereço IP com o valor de 99.130.83.99; indica quais opções estão presentes.

cookie Uma sequência de caracteres que contém algumas informações sobre o cliente, as quais devem ser apresentadas ao servidor sem alterações.

Corporação para Atribuição de Nomes e Números na Internet (ICANN – Internet Corporation for Assigned Names and Numbers) Uma corporação privada sem fins lucrativos gerida por um conselho internacional que assumiu as operações da IANA.

Correção Antecipada de Erros (FEC – Forward Error Correction) correção de erros no receptor sem retransmissão.

corrente contínua (CC) Um sinal de frequência zero e amplitude constante.

correspondência da máscara mais longa (*longest mask matching*) A técnica usada no CIDR pela qual o prefixo mais longo é tratado primeiro no processo de busca em uma tabela de roteamento.

criptografia A ciência e a arte de transformar mensagens para torná-las seguras e imunes a ataques.

criptografia de chave pública Um método de criptografia baseado em um algoritmo que só pode ser invertido usando-se uma informação secreta. O método utiliza dois tipos de chaves: A chave pública é publicamente divulgada, a chave privada (chave secreta) é conhecida apenas pelo seu dono.

criptografia de chave secreta Um conjunto de métodos de segurança em que a chave usada para cifrar é a mesma que a chave para decifrar; tanto o emissor como o receptor têm a mesma chave.

criptografia de chave simétrica Uma cifra na qual a mesma chave é utilizada para cifrar e decifrar.

Curto Espaço entre Quadros (SIFS – Short Interframe Space) No CSMA/CA, um período durante o qual o destino espera após receber o RTS.

dados analógicos Dados que são contínuos e sem variações bruscas, não sendo limitados a um número específico de valores.

dados digitais Dados representados por valores discretos.

dados em rajada Dados que são enviados com diferentes taxas de transmissão a cada instante.

datagrama de usuário O nome do pacote no protocolo UDP.

datagrama IP A unidade de dados do Protocolo Internet.

datagrama No contexto de comutação de pacotes, refere-se a uma unidade de dados independente.

decibel (dB) Uma medida da intensidade relativa de dois sinais.

decifração Recuperação da mensagem original a partir dos dados cifrados.

demodulação O processo de separar o sinal de portadora do sinal que carrega informações.

demodulador Um dispositivo que demodula um sinal modulado para recuperar o sinal original.

demultiplexação O inverso da multiplexação. Processo de obter o sinal/dado original a partir do sinal/dado multiplexado.

desencapsulamento Um processo no qual a carga útil dos pacotes é extraída. Processo inverso ao encapsulamento.

diagrama de constelação Uma representação gráfica da fase e amplitude de diferentes combinações de *bits* em uma modulação digital-analógica.

diagrama de transição de estados Um diagrama para ilustrar os estados de uma máquina de estados finitos.

digitalização Conversão de informação analógica em digital.

dispositivo de conexão Um dispositivo que conecta computadores ou redes.

distância de Hamming mínima Considerando um conjunto de palavras de código, refere-se á menor distância de Hamming entre todos os pares possíveis.

distância de Hamming O número de *bits* diferentes em posições correspondentes em duas palavras de código.

distorção Qualquer alteração em um sinal devido a ruído, atenuação ou outros fatores.

documento ativo (*active document*) Na World Wide Web, refere-se a um documento executado localmente.

documento dinâmico Um documento Web criado como resultado da execução de um programa no lado do servidor.

documento estático Na World Wide Web, refere-se a um documento de conteúdo fixo criado e armazenado em um servidor.

domínio de país Um subdomínio no Sistema de Nomes de Domínio (DNS – Domain Name System) que utiliza dois caracteres como último sufixo (por exemplo, "br", no caso do Brasil).

domínio genérico Um subdomínio no DNS que usa sufixos genéricos.

domínio inverso Um subdomínio do DNS que encontra o nome de domínio dado o endereço IP.

domínio Uma subárvore do espaço de nomes de domínio.

downlink transmissão proveniente de um satélite para uma estação terrestre.

download Operação de recuperar um arquivo ou conjunto de dados provenientes de um local remoto.

DPCM adaptivo (ADPCM – adaptive DPCM) Um método de DPCM no qual o valor de delta é ajustado a cada passo.

Elegibilidade para Descarte (DE – Discard Eligibility) Um *bit* que identifica um pacote que pode ser descartado se houver congestionamento na rede.

elemento de dados A menor entidade que pode representar uma informação. Um *bit*.

elemento de sinal A menor seção de um sinal (temporalmente) que representa um elemento de dados.

embaralhamento (*scrambling*) Na conversão digital-digital, a modificação de parte das regras em uma linha do esquema de codificação para se obter sincronização de *bits*.

encadeamento (*pipelining*) O envio de vários pacotes ou quadros antes de receber informações sobre os anteriores.

encaminhamento de porta Um serviço prestado pelo SSH para permitir que outras aplicações usem canais seguros através do SSH.

Encaminhamento pelo Caminho Reverso (RPF – Reverse Path Forwarding) Uma técnica em que o roteador encaminha somente os pacotes que tenham viajado o caminho mais curto de sua origem até o roteador.

encaminhamento Processo de colocar um pacote em rota para seu destino.

Encapsulamento de Dados de Segurança (ESP – Encapsulating Security Payload) Um protocolo definido pelo IPSec que fornece privacidade e também uma combinação de integridade e autenticação de mensagens.

encapsulamento Técnica na qual uma unidade de dados de um protocolo é colocada dentro do campo de dados da unidade de dados de outro protocolo.

endereçamento com classes (*classful*) Um mecanismo de endereçamento IPv4 em que o espaço de endereços IP é dividido em cinco classes: A, B, C, D e E. Cada classe ocupa uma parte do espaço completo de endereços.

endereçamento sem classes (*classless*) A nova abordagem de endereçamento IPv4 que não utiliza classes com o objetivo de fazer uma melhor utilização do espaço de endereços.

endereço *anycast* Um endereço que define um grupo de computadores, de modo que a mensagem é enviada para o primeiro membro do grupo.

endereço compatível Um endereço IPv6 consistindo de 96 *bits* de zeros seguidos por 32 *bits* do IPv4.

endereço da camada de enlace O endereço de um dispositivo utilizado na camada de enlace de dados (endereço MAC).

endereço de *broadcast* direto O endereço IP em um bloco ou sub-bloco cujo sufixo é formado apenas por 1s. O endereço é usado por roteadores para enviar mensagens para todos os nós daquele (sub-)bloco.

endereço de *broadcast* limitado Um endereço usado para transmitir mensagens de *broadcast* apenas para as estações dentro de uma rede (enlace).

endereço de *broadcast* Um endereço que pode ser utilizado para transmitir uma mensagem para todos os nós de uma rede.

endereço de destino O endereço do receptor de uma unidade de dados.

endereço de enlace local (*link local*) Um endereço IPv6 que é usado se uma LAN deve usar os protocolos Internet, mas não está conectada á Internet por razões de segurança.

endereço de *loopback* Um endereço utilizado por uma estação para testar o seu *software* interno.

endereço de porta Na pilha de protocolos TCP/IP, um número inteiro que identifica um processo (o mesmo que "número de porta").

endereço de rede Um endereço que identifica uma rede para o restante da Internet; ele é o primeiro endereço em um bloco.

endereço de *socket* Uma estrutura contendo um endereço IP e um número de porta.

endereço de sub-rede O endereço de rede de uma sub-rede.

endereço de tratamento (*care-of address*) Um endereço IP temporário utilizado por um *host* móvel enquanto ele se encontra em uma rede visitada.

endereço de tratamento coalocado (*collocated care-of address*) O endereço de tratamento atribuído a um *host* móvel atuando como agente externo.

endereço físico *Ver* endereço da camada de enlace.

endereço Internet Um endereço da camada de rede formado por 32 ou 128 *bits* e empregado para definir univocamente um *host* em uma internet baseada no protocolo TCP/IP.

endereço local No IPv6, refere-se a um endereço para um local com várias redes, mas não conectado à Internet.

endereço lógico Um endereço definido na camada de rede.

Endereço mapeado (*mapped address*) Um endereço IPv6 usado quando um computador que tenha migrado para o IPv6 quer enviar um pacote para um computador que ainda está usando IPv4.

endereço *multicast* Um endereço usado para fazer transmissão *multicast*.

endereço nativo (*home address*) O endereço original de um nó móvel.

endereço *unicast* Um endereço pertencente a um único destino.

enlace (*link*) A via de comunicação física que transfere dados de um dispositivo para outro.

Enlace Assíncrono Sem Conexão (ACL – Asynchronous Connectionless Link) Um canal entre um mestre e um escravo Bluetooth no qual um *payload* (carga útil) danificado é retransmitido.

enlace de *broadcast* Um enlace no qual todas as estações recebem um mesmo pacote enviado.

enlace de trânsito Uma rede com vários roteadores conectados a ela.

enlace *stub* Uma rede que está ligada a apenas um roteador.

enlace virtual Uma conexão entre dois roteadores OSPF que é criada quando o enlace físico está inoperante. O enlace entre eles usa um longo caminho que provavelmente passa por vários roteadores.

enquadramento (*framing*) Agrupamento de um conjunto de *bits* ou de *bytes* em uma unidade de dados.

entrega de pacotes da origem ao destino A transmissão de uma mensagem do remetente original para o destinatário.

entrega direta Entrega feita quando o destino final do pacote é uma máquina conectada á mesma rede física que o remetente daquele pacote.

entrega indireta Uma entrega na qual a origem e o destino de um pacote estão em redes diferentes.

entrega nó a nó Transferência de unidades de dados de um nó para o próximo.

entrega processo a processo A entrega de um pacote proveniente de um processo emissor para o processo receptor.

entrega salto a salto Transmissão de quadros de um nó para o seguinte.

envenenamento reverso (*poisoned reverse*) Uma variação da técnica de horizonte dividido. Nesse método, as informações recebidas pelo roteador são usadas para atualizar a tabela de roteamento e em seguida passadas para todas as interfaces. No entanto, uma entrada na tabela que veio por uma interface é fixada com uma métrica de valor infinito quando ela sai pela mesma interface.

Equipamento nas Instalações do Cliente (CPE – Customer Premises Equipment) No WiMAX, refere-se ao equipamento nas instalações do cliente/assinante, cumprindo o mesmo papel que um modem desempenha na comunicação em redes com fios.

erro de quantização Erro introduzido no sistema durante a quantização (conversão analógico-digital).

erro de um único *bit* Erro em uma unidade de dados no qual um único *bit* é alterado.

erro em rajada Erro em uma unidade de dados em que dois ou mais *bits* são alterados.

espaço de endereçamento O número total de endereços usados por um protocolo.

espaço de nomes de domínio Um método para organizar o espaço de nomes, de modo que eles são definidos em uma estrutura de árvore invertida, com a raiz no topo.

espaço de nomes simples (*flat*) Um espaço de nomes em que não há uma estrutura hierárquica.

espaço de nomes Todos os nomes atribuídos a equipamentos na Internet.

Espaço entre Quadros (IFS – Interframe Space) Em LANs sem fio, um intervalo de tempo entre dois quadros, usado para controlar o acesso ao canal.

Espaço entre Quadros DCF (DIFS – DCF Interframe Space) Em LANs sem fio, um período de tempo que uma estação aguarda antes de enviar um quadro de controle.

Espalhamento Espectral (SS – Spread Spectrum) Uma técnica de transmissão sem fio que requer uma largura de banda igual a várias vezes a largura de banda original.

Espalhamento Espectral por Saltos em Frequências (FHSS – Frequency-Hopping Spread Spectrum) Um método de transmissão sem fios no qual o emissor transmite usando uma frequência portadora por um curto período, em seguida salta para uma outra frequência portadora pela mesma quantidade de tempo, e então salta novamente pela mesma quantidade de tempo, e assim por diante. Depois de *N* saltos, o ciclo é repetido.

Espalhamento Espectral por Sequência Direta (DSSS – Direct Sequence Spread Spectrum) Um método de transmissão sem fio no qual cada *bit* a ser enviado pelo remetente é substituído por uma sequência de *bits* conhecida como código da ficha (*chip code*).

Espalhamento Espectral por Sequência Direta de Alta Taxa (HR-DSSS – High-Rate Direct-Sequence Spread Spectrum) Um método de geração de sinais semelhante ao DSSS exceto pelo método de codificação (CCK).

espectro A faixa de frequências de um sinal.

espectro eletromagnético A faixa de frequência ocupada por energia eletromagnética.

espinha dorsal *multicast* (MBONE – Multicast Backbone) Um conjunto de roteadores da Internet que suportam *multicast* por meio do uso de tunelamento.

estabelecimento de conexão A configuração preliminar necessária para criar uma conexão lógica antes de efetivamente transferir dados.

estação fixa Um *host* que permanece ligado a uma rede.

estação móvel Um nó que pode se mover de uma rede para outra.

estação primária Em um método de acesso primário/secundário, uma estação que envia comandos para as estações secundárias.

estação secundária No método de acesso por varredura/seleção, uma estação que envia uma resposta a um comando proveniente de uma estação primária.

esteganografia Uma técnica de segurança na qual uma mensagem é escondida cobrindo-a com outra coisa.

Estrutura de Gerenciamento da Informação (SMI – Structure of Management Information) No SNMP, um componente usado na gerência de redes.

Ethernet 10-Gigabit Uma nova implementação do Ethernet, operando a 10 Gbps.

Ethernet comutada *full-duplex* Ethernet na qual cada estação, estando em seu próprio domínio de colisão, pode enviar e receber.

Ethernet Gigabit Ethernet com uma taxa de transferência dados de um *gigabit* por segundo (1000 Mbps).

Ethernet Padrão A Ethernet original operando a 10 Mbps.

Ethernet Uma rede de área local criada pela Xerox e que passou por quatro gerações.

Extensões Multifunção para Mensagens de Internet (MIME – Multipurpose Internet Mail Extensions) Um suplemento do SMTP que permite que dados não ASCII sejam enviados via SMTP.

Extensões Seguras Multifunção para Mensagens de Internet (S/MIME – Secure/Multipurpose Internet Mail Extensions) Um aperfeiçoamento do MIME projetado para fornecer segurança para correio eletrônico (e-mail).

extranet Uma rede privada que usa a pilha de protocolos TCP/IP para permitir acesso autorizado de usuários externos.

fase A posição relativa de um sinal no tempo.

fase de configuração Na comutação de circuito virtuais, indica uma fase na qual a origem e o destino usam seus endereços globais para ajudar o *switch* a criar as entradas da tabela referentes à conexão.

fase de finalização Na comutação de circuitos virtuais, a fase na qual a origem e o destino informam ao *switch* que remova a entrada relativa a eles.

fase de transferência de dados A fase intermediária em redes de comutação de circuitos ou de circuitos virtuais, na qual a transferência de dados ocorre.

Fast Ethernet Ethernet com uma taxa de transferência de dados de 100 Mbps.

fator de escala da janela Uma opção no TCP que permite aumentar o tamanho da janela definido no cabeçalho.

fator de reuso No contexto de telefonia celular, refere-se ao número de células com um conjunto diferente de frequências.

FDMA Intercalado (IFDMA – interleaved FDMA) O FDMA mais eficiente usado no Sistema Universal de Telecomunicações Móveis (UMTS – Universal Mobile Telecommunications System).

fibra de modo único Uma fibra óptica com um diâmetro extremamente pequeno que limita os raios de luz a alguns ângulos, resultando em um feixe quase horizontal.

fibra multimodo de índice gradual Uma fibra óptica com um núcleo que possui índice de refração gradual.

fibra óptica Um segmento fino de vidro ou outro material transparente usado para transportar feixes de luz.

fibras multimodo de índice degrau Uma fibra óptica com um núcleo que possui índice de refração uniforme. O índice de refração muda subitamente no limite núcleo/revestimento.

Ficha (*chip*) No CDMA, refere-se a um número correspondente a um código que é atribuído a uma estação.

ficha (*token*) Um pequeno pacote usado no método de acesso de passagem de ficha.

fila de prioridade Uma técnica de filas na qual existem duas filas: uma para pacotes regulares e outra para pacotes com prioridade.

fila FIFO (*First-In, First-Out*, ou Primeiro a Entrar, Primeiro a Sair) Uma fila na qual o primeiro elemento a entrar é o primeiro elemento a sair.

fila justa ponderada Uma técnica de escalonamento de pacotes para melhorar a QoS, na qual cada pacotes é alocado a uma fila com base em um número de prioridade a ele atribuído.

fila Uma lista de espera.

filtragem Processo no qual um *switch* toma decisões de encaminhamento de quadros.

***firewall* de filtro de pacotes** Um *firewall* que bloqueia ou encaminha pacotes com base nas informações dos cabeçalhos da camada de rede e da camada de transporte.

firewall Um dispositivo (comumente um roteador) instalado entre a rede interna de uma organização e o restante da Internet para proporcionar segurança.

fluxo contínuo (*streaming*) de áudio/vídeo armazenado Dados obtidos como arquivos da Internet que o usuário pode escutar ou assistir enquanto ainda não obteve os dados completos.

fluxo contínuo (*streaming*) de áudio/vídeo Transmissão de dados da Internet que o usuário pode escutar ou assistir enquanto ainda não obteve os dados completos.

Folha de Estilo em Cascata (CSS – Cascading Style Sheets) Um padrão desenvolvido para uso com HTML para definir a apresentação do documento.

Força-Tarefa de Engenharia da Internet (IETF – Internet Engineering Task Force) Um grupo de trabalho cujo foco é o projeto e desenvolvimento da pilha de protocolos TCP/IP e da Internet.

Força-Tarefa de Pesquisa da Internet (IRTF – Internet Research Task Force) Um fórum de grupos de trabalho com foco em temas de pesquisa de longo prazo relacionados com a Internet.

Forma de Backus-Naur (BNF – Backus-Naur Form) Uma metalinguagem que especifica quais sequências de símbolos formam um termo válido.

fragmentação A divisão de um pacote em unidades menores para adequação à MTU de um protocolo.

frequência fundamental A frequência da onda senoidal dominante em um sinal composto.

frequência O número de ciclos por segundo de um sinal periódico.

Função de Coordenação Distribuída (DCF – Distributed Coordination Function) O método de acesso básico em LANs sem fio; estações disputam umas com as outras pelo acesso ao canal.

Função de Coordenação Pontual (PCF – Point Coordination Function) Nas LANs sem fio, um método de acesso opcional e complexo implementado em uma rede infraestruturada.

função de *hash* criptográfico Uma função que cria uma saída de tamanho fixo e curto a partir de uma entrada de tamanho arbitrário. Para ser útil, a função deve ser resistente a ataques de pré-imagem, segunda pré-imagem e colisão.

função de *hash* Um algoritmo que cria um resumo criptográfico de tamanho fixo a partir de uma mensagem de comprimento arbitrário.

gatekeeper No padrão H.323, refere-se a um servidor na LAN que desempenha o papel do servidor de registro.

***gateway* de aplicação (*application gateway*)** Em um *firewall proxy*, o computador que fica entre o computador-cliente e o computador da corporação.

Globalstar Um sistema de satélites LEO com 48 satélites em seis órbitas polares, sendo que cada órbita apresenta oito satélites.

Grupo de Especialistas em Fotografia (JPEG – Joint Photographic Experts Group) Um padrão para compressão de imagens compostas por tonalidades contínuas.

Grupo de Especialistas em Imagens em Movimento (MPEG – Motion Picture Experts Group) Um método para compressão de vídeo.

grupo de trabalho (*working group*) Um comitê da IETF concentrado em um tópico específico da Internet.

H.323 Um padrão projetado pela ITU para permitir que telefones da rede pública de telefonia se comuniquem com computadores (os chamados terminais H.323) conectados à Internet.

harmônicos Componentes de um sinal digital, cada qual tendo uma amplitude, frequência e fase diferentes.

hertz (Hz) Unidade de medida de frequência.

Hierarquia Digital Síncrona (SDH – Synchronous Digital Hierarchy) A equivalente, proposta pela ITU-T, à tecnologia SONET.

hipermídia Informação contendo texto, imagens, gráficos e som que estão ligados a outros documentos por meio de ponteiros (*links*).

hipertexto (*hypertext*) Informação contendo texto que está ligado a outros documentos por meio de ponteiros (*links*).

horizonte dividido Um método para melhorar a estabilidade do RIP no qual o roteador seletivamente escolhe a interface a partir da qual a informação de atualização é enviada.

host Uma estação ou nó que faz parte de uma rede.

hub Um dispositivo central em uma topologia em estrela, provendo uma conexão compartilhada pelos nós.

ID de *host* (*hostid*) A parte de um endereço IP que identifica um *host*.

identificador de objeto (oid – object identifier) No contexto de uma MIB, refere-se a um identificador para um objeto usado no SNMP e alguns outros protocolos de gerenciamento de rede.

identificador de rede (netid) A parte de um endereço IP que identifica a rede.

impasse (*deadlock*) Uma situação de impasse, na qual uma tarefa não pode prosseguir porque está esperando por um evento que nunca ocorrerá.

Índice do Parâmetro de Segurança (SPI – Security Parameter Index) Um parâmetro que distingue univocamente uma associação de segurança das outras.

Infraestrutura de Chaves Públicas (PKI – Public Key Infrastructure) A estrutura hierárquica dos servidores de uma Autoridade Certificadora (AC).

Instituto de Engenheiros Eletricistas e Eletrônicos (IEEE – Institute of Electrical and Electronics Engineers) Um grupo formado por engenheiros profissionais. O grupo tem sociedades especializadas cujos comitês elaboram normas nas áreas de especialidade de seus membros.

Instituto Nacional Americano de Padrões (ANSI – American National Standards Institute) Uma organização nacional de normalização que define padrões nos Estados Unidos.

intercalamento (*interleaving*) No contexto de multiplexação, refere-se à técnica de tomar uma quantidade específica de dados de cada dispositivo em uma ordem regular.

interconexão Ato de conectar várias redes entre si usando dispositivos de interconexão, como roteadores e *gateways*.

interface A fronteira entre dois equipamentos. Também se refere a características mecânicas, elétricas e funcionais da conexão. No contexto da programação de redes, designa um conjunto de funções disponíveis para que a camada superior utilize os serviços da camada inferior.

Interface de Camada de Transporte (TLI – Transport Layer Interface) Uma API de rede fornecida pelo sistema UNIX.

Interface de Comunicação Comum (CGI – Common Gateway Interface) Um padrão para comunicação entre servidores HTTP e programas executáveis. A CGI é utilizada na criação de documentos dinâmicos.

Interface de Programação de Aplicativos (API – Application Programming Interface) Um conjunto de declarações, definições e procedimentos seguidos pelos programadores para escrever programas cliente-servidor.

interface de *socket* Um conjunto de chamadas de sistema utilizadas para programação de aplicações de rede.

Interface Rede-Rede (NNI – Network-to-Network Interface) No ATM, uma interface entre duas redes.

Interface Usuário-Rede (UNI – User-to-Network Interface) No ATM, a interface entre um ponto final (usuário) e um *switch* ATM.

interferência cruzada (diafonia) O ruído em uma linha causado por sinais que viajam por uma outra linha.

interferência Qualquer energia indesejada que interfere com os sinais desejados.

Internet draft Um documento na Internet que representa um trabalho em andamento, sem qualquer *status* oficial e uma vida útil de seis meses.

***Internet Engineering Steering Group* (IESG)** Uma organização que supervisiona as atividades da IETF.

internet Um conjunto de redes conectadas por dispositivos de interconexão, como roteadores ou *gateways*.

Internet Uma internet mundial que usa o conjunto de protocolos TCP/IP.

***internetwork* (ou internet)** Um conjunto de redes interligadas.

Interoperabilidade Mundial para Acesso por Micro-ondas (WiMAX – Worldwide Interoperability for Microwave Access) Uma família de padrões IEEE 802.16 para transmissão de dados sem fio na última milha (similar a redes a cabo ou DSL para a comunicação com fio).

intranet Uma rede privada que usa a pilha de protocolos TCP/IP.

inundação (*flooding*) Saturação de uma rede com mensagens.

Inversão Alternada de Marcas (AMI – Alternate Mark Inversion) Método de codificação bipolar digital-digital no qual a amplitude que representa o valor 1 alterna entre voltagens positivas e negativas.

IPSec (IP *Security*, ou Segurança IP) Um conjunto de protocolos projetados pela IETF (*Internet Engineering Task Force*) para prover segurança a pacotes trafegando na Internet.

Iridium Uma rede de 66 satélites que fornece comunicação entre quaisquer dois pontos da Terra.

irretratabilidade Um aspecto de segurança no qual um receptor deve ser capaz de provar que uma mensagem recebida veio de um remetente específico.

ISP regional Uma pequeno ISP que fica conectado a um ou mais *backbones* ou ISPs internacionais.

janela de contenção No CSMA/CA, refere-se à divisão de uma certa quantidade de tempo em pequenos intervalos discretos (*slots*). Cada intervalo é selecionado aleatoriamente por uma estação para a transmissão.

janela deslizante Um método que permite que diversas unidades de dados sejam transmitidas antes que o emissor receba uma confirmação.

Java Uma linguagem de programação orientada a objetos.

***jitter* (variação de atraso)** Um fenômeno de transmissões em tempo real causado por atrasos irregulares, levando a lacunas entre pacotes consecutivos no receptor.

Kademlia Uma rede P2P baseada em DHT na qual as distâncias entre os nós são medidas por meio de uma operação de OU-Exclusivo (XOR) entre dois identificadores.

lacete local (*local loop*) Também conhecido como rede de última milha, é a conexão física local que liga o assinante à central de telefonia.

largura de banda A diferença entre a maior e a menor frequências de um sinal composto. Também mede a capacidade de uma linha ou rede em carregar informações.

leased line Linha dedicada alugada.

Lempel-Ziv-Welch (LZW) Um grupo de métodos de compressão baseados na criação dinâmica de um dicionário (vetor) de sequências de caracteres do texto. A técnica foi inventada por Lempel e Ziv e refinada por Welch.

ligação ascendente (*uplink*) Transmissão de uma estação terrestre para um satélite.

Linguagem de Estilo Extensível (XSL – Extensible Style Language) A linguagem de estilo do XML.

Linguagem de Marcação de Hipertexto (HTML – HyperText Markup Language) A linguagem computacional para especificar conteúdo e formato de um documento Web. Ela permite textos adicionais que incluem códigos para definir fontes, *layouts*, gráficos incorporados e atalhos de hipertexto.

Linguagem de Marcação de Hipertexto Extensível (XHTML – Extensible HyperText Markup Language) HTML em conformidade com a sintaxe do XML.

Linguagem de Marcação Extensível (XML – Extensible Markup Language) Uma linguagem que permite aos usuários definir a representação dos dados.

Linha de Assinante Digital (DSL – Digital Subscriber Line) Uma tecnologia que usa redes de telecomunicações existentes para prover entrega de dados, voz, vídeo e multimídia em alta velocidade.

Linha de Assinante Digital Adaptativa a Taxa (RADSL – Rate Adaptive Asymmetrical Digital Subscriber Line) Uma tecnologia baseada em DSL que apresenta diferentes taxas de transmissão de dados, dependendo do tipo de comunicação.

linha de estado Na mensagem de resposta HTTP, é uma linha que contém a versão de HTTP, um espaço, um código de estado, um espaço e uma frase de estado.

linha T Uma hierarquia de linhas digitais projetadas para transportar voz e outros sinais em formas digitais.

Localizador Uniforme de Recursos (URL – Uniform Resource Locator) Uma sequência de caracteres (endereço) que identifica uma página na World Wide Web.

LSA de enlace resumo para rede Um pacote LSA, que determina o custo para alcançar redes externas à área.

LSA de enlace resumo para roteador de borda do AS Um pacote LSA que permite que um roteador dentro de uma área aprenda a rota para um roteador de borda autônomo.

MAC baseado em *hash* (HMAC – hashed MAC) Um algoritmo de MAC baseado em uma função *hash*, como o SHA-1.

mapeamento dinâmico Uma técnica na qual um protocolo é usado para resolução de endereços.

mapeamento estático Uma técnica na qual uma lista de correspondências entre endereços lógicos e físicos é usada para a resolução de endereços.

Máquina de Estados Finitos (MEF) Uma máquina que passa por um número limitado de estados.

marcador (*flag*) Uma sequência de *bits* ou um caractere adicionado ao início e ao fim de um quadro para separar os quadros.

máscara de sub-rede A máscara para uma dada sub-rede.

máscara de super-rede A máscara de uma super-rede.

máscara No IPv4, um número binário de 32 *bits* que fornece o primeiro endereço do bloco (o endereço de rede) quando aplicado a um endereço IP (outro número binário de 32 *bits*).

mascaramento de frequência (*frequency masking*) Ocorre quando um som alto mascara parcial ou totalmente um som mais baixo, caso as frequências dos dois sejam próximas.

mascaramento temporal Uma situação em que um som alto pode amortecer os nossos ouvidos por um tempo curto, mesmo após o som parar.

mecanismo de carona (*piggybacking*) A inclusão de uma confirmação juntamente com um quadro de dados.

Mecanismo Seguro de Troca de Chaves (SKEME – Secure Key Exchange Mechanism) Um protocolo para troca de chaves, desenvolvido por Hugo Krawcyzk, que usa criptografia de chave pública para autenticação de entidades.

meio de transmissão O caminho físico que liga dois dispositivos de comunicação.

meio guiado Meio de transmissão que apresenta um limite físico.

meio não guiado Meios de transmissão sem limites físicos (ar).

Memória Apenas de Leitura (ROM – Read-Only Memory) Memória permanente, cujo conteúdo não pode ser alterado.

Menor Rota Primeiro Aberto (OSPF – Open Shortest Path First) Um protocolo de roteamento interno baseado no roteamento por estado de enlace.

metarquivo No contexto de transmissão de áudio ou vídeo em fluxo contínuo, um arquivo que contém as informações sobre o arquivo de áudio/vídeo.

método 1-persistente Uma estratégia de persistência do CSMA na qual uma estação envia um quadro imediatamente se o meio estiver livre.

método não persistente Um método de acesso aleatório múltiplo em que uma estação espera um período de tempo aleatório após uma colisão ser detectada.

método p-persistente Uma estratégia de persistência do CSMA na qual uma estação envia dados com uma probabilidade p se encontrar a linha ociosa.

métrica Um custo atribuído a algum caminho em uma rede.

micro-onda As ondas eletromagnéticas que se encontram na faixa de 2 GHz a 40 GHz.

MIMO multiusuário (MU-MIMO – Multi-User MIMO) Uma versão mais sofisticada do MIMO no qual múltiplos usuários podem se comunicar ao mesmo tempo.

misturador (*mixer*) Um dispositivo que soma matematicamente os sinais provenientes de fontes diferentes para criar um único sinal.

mobilidade de transição No IEEE 802.11, uma estação com mobilidade de transição inter-BSS pode se mover de um BSS para outro, mas o movimento fica confinado dentro de um ESS. Uma estação de transição sem mobilidade ou é estacionária (não móvel) ou está movendo-se apenas dentro de um BSS.

Mobilidade do tipo transição inter-BSS Em uma LAN sem fio, uma estação que pode se mover de um BSS para outro, mas está confinada dentro de um ESS.

mobilidade sem transição *Ver* mobilidade com transição.

Modelo de Interconexão de Sistemas Abertos (OSI – Open Systems Interconnection) Um modelo de sete camadas definido pela ISO para comunicação de dados.

modelo Internet Uma pilha de protocolos contendo cinco camadas, que domina as redes e as comunicações de dados atuais.

modem a cabo (*cable modem*) Uma tecnologia na qual a TV a cabo provê acesso à Internet.

modem de 56K Uma tecnologia de modem usando duas taxas de transferência de dados distintas: uma para *upload* e outra para *download* na Internet.

modem Um dispositivo constituído por um modulador e um demodulador. Ele converte um sinal digital em um sinal analógico (modulação) e vice-versa (demodulação).

Modo Balanceado Assíncrono (ABM – Asynchronous Balanced Mode) No HDLC, refere-se a um modo de comunicação no qual cada estação pode ser primária ou secundária.

Modo de Resposta Normal (NRM – Normal Response Mode) No HDLC, um modo de comunicação no qual a estação secundária deve ter permissão da estação primária antes de a transmissão poder prosseguir.

Modo de Transmissão Assíncrono (ATM – Asynchronous Transfer Mode) Um protocolo de longa distância apresentando altas taxas de transmissão de dados e pacotes de tamanhos iguais (células); o ATM é adequado para a transferência de dados de texto, áudio e vídeo.

modo de transporte Modo de operação do IPSec em que um segmento TCP ou datagrama de usuário UDP é cifrado e então encapsulado em um pacote IP.

modo *full-duplex* Um modo de transmissão no qual ambas as partes podem se comunicar simultaneamente.

modo half-duplex Um modo de transmissão em que a comunicação pode ser bidirecional, mas os dois lados não podem transmitir ao mesmo tempo.

modo simplex Um modo de transmissão em que a comunicação é unidirecional.

modo túnel Um modo no IPSec que protege o pacote IP inteiro. Ele pega um pacote IP, incluindo o cabeçalho, aplica métodos de segurança do IPSec a todo o pacote e em seguida adiciona um novo cabeçalho IP.

modulação Alteração de uma ou mais características de uma onda portadora usando um sinal que carrega informações.

Modulação de Amplitude de Pulso (PAM – Pulse Amplitude Modulation) Uma técnica na qual um sinal analógico é amostrado; o resultado é uma série de pulsos baseados nos dados amostrados.

Modulação de Amplitude de Pulso de 5 Níveis e 4 Dimensões (4D-PAM5 – 4-Dimensional, 5-Level Pulse Amplitude Modulation) Um esquema de codificação usado pelo padrão 1000Base-T.

Modulação de Fase (PM – Phase Modulation) Um método de modulação analógico-analógico no qual a fase do sinal de portadora varia com a amplitude da modulação do sinal.

Modulação de Frequência (FM – Frequency Modulation) Um método de modulação analógico-analógico em que a frequência do sinal da portadora varia com a amplitude do sinal de modulação.

Modulação Delta Adaptativa (ADM – Adaptive Delta Modulation) Uma técnica de modulação delta na qual o valor de delta é ajustado a cada passo.

modulação delta Uma técnica de conversão analógico-digital na qual o valor do sinal digital baseia-se na diferença entre o valor recém-amostrado e o valor anterior.

modulação em amplitude (AM – amplitude modulation) Um método de conversão analógico-analógico em que a amplitude do sinal da portadora varia de acordo com a amplitude do sinal de modulação.

Modulação por Amplitude em Quadratura (QAM – Quadrature Amplitude Modulation) Um método de modulação digital-analógico no qual a fase e a amplitude do sinal de portadora variam com o sinal de modulação.

Modulação por Chaveamento de Amplitude (ASK – Amplitude Shift Keying) Um método de conversão digital-analógico em que a amplitude do sinal da portadora varia para representar o valor binário 0 ou 1.

Modulação por Chaveamento de Fase (PSK – Phase Shift Keying) Um método de modulação digital-analógico no qual a fase do sinal de portadora é variada para representar um padrão de *bits* específico.

Modulação por Chaveamento de Frequência (FSK – Frequency Shift Keying) Um método de conversão digital-analógico em que a frequência do sinal da portadora varia para representar o valor binário 0 ou 1.

Modulação por Código de Pulsos (PCM – Pulse Code Modulation) Uma técnica que modifica pulsos PAM para criar um sinal digital.

Modulação por Posição de Pulso (PPM – Pulse Position Modulation) A técnica de modulação utilizada para modular um sinal infravermelho.

modulador Um dispositivo que modula um sinal para criar outro sinal.

módulo O limite superior na aritmética modular (n).

***Multicast* Independente de Protocolo (PIM – Protocol Independent Multicast)** A família de protocolos de *multicast* com dois membros, PIM-DM e PIM-SM; ambos os protocolos são dependentes do protocolo de *unicast*.

***Multicast* Independente de Protocolo-Modo Denso (PIM-DM – Protocol Independent Multicast-Dense Mode)** Um protocolo de roteamento baseado na origem que usa RPF e estratégias de poda/enxerto para tratar o *multicast*.

***Multicast* Independente de Protocolo-Modo Esparso (PIM-SM – Protocol Independent Multicast-Sparse Mode)** Um protocolo de roteamento de grupo compartilhado que é semelhante ao CBT e utiliza um ponto de encontro (*rendezvous point*) como a origem da árvore.

***Multicast* pelo Caminho Reverso (RPM – Reverse Path Multicasting)** Uma técnica que adiciona mecanismos de poda e enxerto ao RPB para criar uma árvore *multicast* de caminho mais curto com suporte a alterações dinâmicas dos membros.

multicast Um método de transmissão que permite que as cópias de um único pacote sejam enviadas para um grupo de receptores selecionados.

multiplexação densa por comprimento de onda (DWDM – dense wave-division multiplexing) Um método de WDM que pode multiplexar um número muito grande de canais em uma mesma fibra devido ao menor espaçamento entre os canais.

multiplexação O processo de combinar sinais de várias fontes para transmissão por meio de um único enlace de dados.

Multiplexação por Divisão de Comprimento de Onda (WDM – Wavelengh-Division Multiplexing) A combinação de sinais de luz modulados em um único sinal.

Multiplexação por Divisão de Frequência (FDM – Frequency-Division Multiplexing) A combinação de sinais analógicos em um único sinal.

Multiplexação por Divisão de Frequências Ortogonais (OFDM – Orthogonal Frequency Division Multiplexing) Um método de multiplexação semelhante ao FDM, com todas as sub-bandas usadas por um emissor em um determinado momento.

Multiplexação por Divisão de Tempo (TDM – Time-Division Multiplexing) A técnica de combinar os sinais provenientes de canais de baixa velocidade para compartilhar o tempo em um caminho de alta velocidade.

multiplexador (MUX) Um dispositivo usado para multiplexação.

Multiplexador de Acesso à Linha de Assinante Digital (DSLAM – Digital Subscriber Line Access Multiplexer) Um dispositivo instalado na companhia de telefonia que funciona como um modem ADSL.

multiplexador de inserção/remoção (*add/drop multiplexer*) Dispositivo SONET que remove e insere sinais em um caminho sem que haja demultiplexação e remultiplexação.

multiplexador/demultiplexador STS Um dispositivo SONET que multiplexa e demultiplexa sinais.

Não Retorno ao Zero (NRZ – Non-Return-to-Zero) Um método de codificação polar digital-digital no qual o nível do sinal é sempre positivo ou negativo, mas nunca fica no nível zero.

Não Retorno ao Zero, Inversão (NRZ-I – Non-return-to-Zero, Invert) Um método de codificação NRZ em que o nível do sinal é invertido cada vez que um 1 é encontrado.

Não Retorno ao Zero, Nível (NRZ-L – Non-Return-to-Zero, Level) Um método de codificação NRZ em que o nível de sinal está diretamente relacionado com o valor do *bit*.

navegador Web (*browser*) Um aplicativo que exibe documentos WWW. Um navegador Web normalmente usa outros serviços da Internet para acessar o documento.

negação de serviço (DoS – denial of service) Ataque contra a meta de disponibilidade de um sistema, podendo deixá-lo lento ou inacessível.

nível de maturidade As fases através da qual uma RFC passa.

nó Um dispositivo de comunicação endereçável (por exemplo, um computador ou roteador) em uma rede.

nome de domínio No DNS, refere-se a uma sequência de rótulos separados por pontos.

Nome de Domínio Parcialmente Qualificado (PQDN – Partially Qualified Domain Name) Um nome de domínio que não inclui todos os níveis entre o *host* e o nó-raiz.

Nome de Domínio Totalmente Qualificado (FQDN – Fully Qualified Domain Name) Um nome de domínio que consiste em identificadores começando pela estação e passando por todos os níveis até o nó-raiz.

nonce Um número aleatório grande utilizado uma vez para distinguir uma nova solicitação de autenticação de uma antiga.

notação de barra Um método para indicar o número de 1s na máscara de forma resumida.

notação decimal com pontos Uma notação concebida para tornar o endereço IP mais fácil de ler; cada *byte* é convertido em seu equivalente decimal e, em seguida, separado de seu vizinho por um ponto.

notação hexadecimal com dois pontos (*colon hexadecimal notation*) No IPv6, refere-se a uma notação que representa endereços usando 32 dígitos hexadecimais com dois pontos separando cada grupo de quatro dígitos.

Notação Sintática Abstrata.1 (ASN.1 – Abstract Syntax Notation One) Uma linguagem formal usando sintaxe abstrata para definir a estrutura de uma Unidade de Dados de Protocolo (PDU – Protocol Data Unit).

núcleo (*core*) O centro de vidro ou de plástico de uma fibra óptica.

Núcleo de Informação e Coordenação (NIC – Network Information Center) Uma agência responsável por coletar e distribuir informações sobre os protocolos TCP/IP.

Núcleo de Informação e Coordenação da Internet (INTERNIC – Internet Network Information Center) Uma agência responsável por coletar e distribuir informações sobre protocolos TCP/IP.

número da porta efêmera Um número de porta usado por aplicações cliente.

número de confirmação (*acknowledgment number*) No TCP, é o número no campo de confirmação que define o número de sequência do próximo *byte* esperado.

número de porta bem conhecido Um número de porta que identifica um processo no servidor.

número de porta Um número inteiro que define um processo sendo executado em um *host* (o mesmo que "endereço de porta").

Número de Sequência Inicial (ISN – Initial Sequence Number) No TCP, refere-se ao número aleatório usado como o primeiro número de sequência em uma conexão.

número de sequência O número que indica a localização de um quadro ou pacote em uma mensagem.

Oakley Um protocolo de estabelecimento de chave, desenvolvido por Hilarie Orman, que é um dos três componentes do protocolo IKE.

onda de rádio Energia eletromagnética na faixa de 3 KHz a 300 GHz.

onda senoidal Uma representação de amplitude em função do tempo de um vetor rotativo.

operadora comum (*common carrier*) Uma instalação de comunicação disponível ao público e cuja utilização é sujeita à regulamentação pública.

Órbita Geoestacionária (GEO – Geostationary Earth Orbit) Uma órbita do satélite localizada acima do cinturão superior de Van Allen. Um satélite viajando nessa órbita parece estar em uma posição fixa para pessoas na Terra.

órbita O caminho que um satélite percorre ao redor da Terra.

Órbita Terrestre Baixa (LEO – Low-Earth-Orbit) Uma órbita de satélites polares com uma altitude entre 500 e 2.000 km. Um satélite com essa órbita tem um período de rotação de 90 a 120 minutos.

Órbita Terrestre Média (MEO – Medium-Earth-Orbit) Uma órbita de satélites posicionada entre os dois cinturões de Van Allen. Um satélite nessa órbita leva seis horas para circundar a Terra.

Organização Internacional de Padronização (ISO – nternational Organization for Standardization) Uma organização internacional que define e desenvolve padrões em uma ampla variedade de tópicos.

origem extinta (*source quench*) Um método usado no ICMP para controle de fluxo, no qual a origem é aconselhada a desacelerar ou interromper o envio de datagramas devido a congestionamento.

overhead Carga computacional adicional introduzida por um processo ou protocolo.

pacote de bloqueio (*choke*) Um pacote enviado por um roteador para a origem da transmissão para informá-lo da existência de congestionamento.

Pacote de Estado de Enlace (LSP – Link-State Packet) No contexto de roteamento baseado no estado de enlace (*link-state*), refere-se a um pequeno pacote contendo informações de roteamento enviadas por um roteador para todos os outros roteadores.

pacote Sinônimo para unidade de dados, usado principalmente em descrições na camada de rede.

Padrão Avançado de Cifração (AES – Advanced Encryption Standard) Uma cifra de bloco assíncrona adotada pelo NIST para substituir o DES.

Padrão de Assinatura Digital (DSS – Digital Signature Standard) O padrão de assinatura digital adotado pelo NIST sob a norma FIPS 186.

Padrão de Cifração de Dados (DES – Data Encryption Standard) Uma cifra de bloco simétrica baseada na estrutura de Feistel e padronizada pelo NIST.

padrão Internet Uma especificação exaustivamente testada útil e aceita por aqueles que trabalham com a Internet. Constitui uma regulamentação formal que deve ser seguida.

Padrão Provisório 95 (IS-95 – Interim Standard) Um dos padrões de telefonia celular de segunda geração dominantes na América do Norte.

página Web Uma unidade de hipertexto ou hipermídia disponível na Web.

palavra de código (*codeword*) Uma palavra de dados (*dataword*) codificada.

palavra de dados (*dataword*) O menor conjunto de dados usado em codificação de bloco.

Par Trançado Blindado (STP – Shielded Twisted-Pair) Cabo de par trançado revestido por uma folha ou malha blindada que protege contra interferências eletromagnéticas.

Par Trançado sem Blindagem (UTP – Unshielded Twisted-Pair) Um cabo com fios que são torcidos juntos para reduzir ruído e interferências. *Ver* também ***cabo de par trançado*** e ***par trançado com blindagem***.

paradigma cliente-servidor Um paradigma em que dois computadores são conectados por uma internet e cada um deve executar um programa, um deles para fornecer e o outro para solicitar um serviço.

paradigma *peer-to-peer* (P2P) Também conhecido como par a par, um paradigma no qual computadores de usuários finais podem se comunicar entre si para trocar serviços.

partida lenta Um método de controle de congestionamento no qual o tamanho da janela de congestionamento inicialmente aumenta exponencialmente.

passagem de ficha Um método de acesso no qual uma ficha é circulada na rede. A estação que recebe e retém a ficha pode enviar dados.

Pastry Uma rede P2P baseada em DHT na qual os identificadores são conjuntos de caracteres de n-dígitos na base 2^b.

payload Carga útil carregada em uma mensagem.

PCM diferencial (DPCM – differential PCM) O DPCM é a generalização da modulação delta na qual mais de uma amostra previamente reconstruída é usada para a predição da próxima amostra.

Pedido de Comentários (RFC – Request for Comment) Um documento formal da Internet relativo a um assunto da Internet.

pedido de repetição automática (ARQ – automatic repeat request) Um método de controle de erros no qual a correção é feita por meio da retransmissão de dados.

período A quantidade de tempo necessária para um sinal completar um ciclo completo.

piconet Um tipo de rede Bluetooth.

pilha de protocolos TCP/IP Um grupo de protocolos hierárquicos usados em uma internet (por exemplo, na Internet)

pilha de protocolos Uma pilha ou família de protocolos definidos para um sistema de comunicações complexo.

pilha dupla (*dual stack*) Dois protocolos (IPv4 e IPv6) em uma mesma estação.

pixel Considerado o menor componente de uma imagem digital.

placa de interface de rede (NIC – network interface card) Um dispositivo eletrônico (interno ou externo) a uma estação, que contém circuitos que permitem que a estação possa ser conectada à rede.

plotagem do domínio de frequências Uma representação gráfica das frequências que compõem um sinal.

poda Ato de interromper o envio de mensagens *multicast* por uma interface.

polinômio Um termo algébrico que pode ser utilizado como divisor no cálculo de um CRC.

política de roteamento Um recurso do roteamento por vetor de caminhos no qual as tabelas de roteamento são baseadas em regras definidas pelo administrador da rede, e não de uma métrica.

Política de Segurança (SP – Security Policy) No IPSec, um conjunto de requisitos de segurança predefinidos aplicados a um pacote quando este estiver pronto para ser enviado ou quando chega.

ponte (*bridge*) Um dispositivo de rede operando nas duas primeiras camadas do modelo Internet, fornecendo recursos de filtragem e encaminhamento.

ponte remota Um dispositivo que conecta redes locais e redes ponto a ponto, muitas vezes usado em redes *backbone*.

ponte simples Um dispositivo de rede que liga dois segmentos; requer manutenção e atualização manuais.

ponto de acesso (AP – access point) Uma estação-base central em um BSS.

Ponto de Acesso à Rede (NAP – Network Access Point) Uma estação de comutação complexa que conecta redes de *backbone*.

ponto de cruzamento (*crosspoint*) O ponto de junção de uma entrada e uma saída em um comutador *crossbar*.

Ponto de Encontro (RP – Rendezvous Point) Um roteador usado pelo PIM para distribuir pacotes *multicast*.

Ponto de Presença (POP – Point of Presence) Uma estação de comutação onde as operadoras podem interagir umas com as outras.

porta IrDA Uma porta que permite que um teclado sem fios se comunique com um computador.

porta registrada Um número de porta, variando de 1.024 a 49.151, não atribuído ou controlado pela IANA.

Portadora Óptica (OC – Optical Carrier) A hierarquia das operadoras de fibra óptica definidas na SONET.

preâmbulo O campo de 7 *bytes* de um quadro IEEE 802.3 consistindo em 1s e 0s alternados que alertam o receptor do envio de dados e permitem a sincronização do receptor.

preenchimento de *bits* (*bit stuffing*) Em um protocolo orientado a *bits*, refere-se ao processo de adicionar um *bit* extra no campo de dados de um quadro para evitar que uma sequência se pareça com um identificador de significado reservado.

preenchimento de *bytes* (*byte stuffing*) Em um protocolo orientado a *bytes*, é o processo de adicionar um *byte* extra no campo de dados de um quadro para evitar que uma sequência se pareça com um identificador de significado reservado.

preenchimento de pulso No TDM, uma técnica que adiciona *bits* extras às linhas de entrada com taxas mais baixas.

prefixo No endereçamento IP, é um outro nome para a parte comum do bloco (similar a *netid*).

Privacidade Bastante Boa (PGP – Pretty Good Privacy) Um protocolo inventado por Phil Zimmermann para prover privacidade, autenticação e integridade ao e-mail.

privacidade Um aspecto de segurança no qual a mensagem só faz sentido para o receptor à qual ela se destina.

processamento distribuído Uma estratégia na qual os serviços fornecidos residem em múltiplos locais.

processo *peer-to-peer* Um processo no qual um equipamento receptor e outro emissor se comunicam em uma determinada camada.

processo Um programa aplicativo em execução.

processo-cliente Um aplicativo sendo executado localmente que solicita serviços de um aplicativo em execução em um local remoto.

Procurador de Pacotes na Internet (PING – Packet Internet Groper) Um programa de aplicação usado para determinar a acessibilidade de um destino usando uma solicitação de eco e resposta ICMP.

produto interno Um número produzido pela multiplicação de duas sequências, elemento a elemento, seguida da soma dos produtos.

produto largura de banda-atraso Uma medida do número de *bits* que podem ser enviados antes que se receba qualquer notificação do receptor.

Projeto 802 O projeto realizado pelo IEEE em uma tentativa de resolver a incompatibilidade de LANs.

propagação celeste Propagação de ondas de rádio para a ionosfera e em seguida de volta à Terra.

propagação em linha de visada (*line-of-sight*) A transmissão em linha reta de sinais de frequência muito alta, diretamente de antena a antena.

propagação no espaço Um tipo de propagação que pode penetrar na ionosfera.

propagação terrestre Propagação de ondas de rádio através da porção mais baixa da atmosfera (junto à Terra).

protocolo Conjunto de regras para a comunicação.

Protocolo de Acesso a Correio da Internet (IMAP – Internet Mail Access Protocol) Um protocolo complexo e poderoso para recuperar mensagens de e-mail de um servidor de e-mail.

Protocolo de Adaptação e Controle de Enlace Lógico (L2CAP – Logical Link Control and Adaptation Protocol) Uma camada Bluetooth usada para a troca de dados em um enlace assíncrono sem conexão (ACL)

Protocolo de Agência de Correio versão 3 (POP3 – Post Office Protocol version 3) Um protocolo de acesso a e-mail popular, mas simples.

protocolo de apresentação (*handshake*) O protocolo utilizado em redes orientadas à conexão para estabelecer ou encerrar uma conexão.

Protocolo de Autenticação por Senha (PAP – Password Authentication Protocol) Um protocolo de autenticação simples de duas etapas usado no PPP.

protocolo de Cabeçalho de Autenticação (AH – Authentication Header) Um protocolo definido pelo IPSec na camada de rede que fornece integridade às mensagens por meio da criação de uma assinatura digital com uma função de *hash*.

protocolo de Camada de *Sockets* Segura (SSL – Secure Sockets Layer) Um protocolo projetado para fornecer serviços de segurança e compressão para dados provenientes da camada de aplicação.

Protocolo de Configuração Dinâmica de *Host* (DHCP – Dynamic Host Configuration Protocol) Uma extensão do BOOTP que atribui informações de configuração dinamicamente aos *hosts*.

Protocolo de Controle de Enlace (LCP – Link Control Protocol) Um protocolo PPP responsável pelo estabelecimento, manutenção, configuração e encerramento de enlaces.

Protocolo de Controle de Fluxo de Transmissão (SCTP – Stream Control Transmission Protocol) O protocolo da camada de transporte projetado para combinar as características do UDP e do TCP.

Protocolo de Controle de Rede (NCP – Network Control Protocol) No PPP, refere-se a um conjunto de protocolos de controle que permitem o encapsulamento de dados provenientes de protocolos da camada de rede.

Protocolo de Controle de Rede da Internet (IPCP – Internet Protocol Control Protocol) No PPP, refere-se ao conjunto de protocolos para estabelecer e encerrar uma conexão da camada de rede para pacotes IP.

Protocolo de Controle de Transmissão (TCP – Transmission Control Protocol) Um protocolo da camada de transporte da pilha de protocolos TCP/IP.

Protocolo de Controle de Transmissão/Protocolo Internet (TCP/IP – Transmission Control Protocol/Internetwork Protocol) Uma pilha de protocolos de cinco camadas que especifica as transmissões de dados na Internet.

Protocolo de Controle de Transporte em Tempo Real (RTCP – Real-time Transport Control Protocol) Um protocolo companheiro do RTP contendo mensagens que controlam o fluxo e a qualidade dos dados, além de permitir ao destinatário enviar informações sobre eventos diversos para a origem (ou origens).

Protocolo de Datagramas de Usuário (UDP – User Datagram Protocol) Um protocolo TCP/IP da camada de transporte não orientado à conexão.

Protocolo de Fluxo Contínuo em Tempo Real (RTSP – Real-Time Streaming Protocol) Um protocolo de controle fora de banda projetado para adicionar maiores funcionalidades aos processos de transmissão em fluxo contínuo (*streaming*) de áudio/vídeo.

Protocolo de Gerenciamento de Chaves e Associações de Segurança da Internet (ISAKMP – Internet Security Association and Key Management Protocol) Um protocolo desenvolvido pela Agência de Segurança Nacional americana (NSA) que implementa as trocas definidas pelo protocolo IKE.

Protocolo de Gerenciamento de Grupos Internet (IGMP – Internet Group Management Protocol) Um protocolo da pilha de protocolos TCP/IP que lida com transmissão via *multicast*.

Protocolo de Informações de Roteamento (RIP – Routing Information Protocol) Um protocolo de roteamento baseado no algoritmo de vetor de distâncias.

Protocolo de Iniciação de Sessão (SIP – Session Initiation Protocol) No contexto de voz sobre IP, refere-se a um protocolo da camada de aplicação que estabelece, gerencia e finaliza uma sessão multimídia.

Protocolo de Inicialização (BOOTP – Bootstrap Protocol) O protocolo que fornece informações de configuração a partir de uma tabela (arquivo).

protocolo de janela deslizante Um protocolo que usa o método de janela deslizante.

Protocolo de Mensagens de Controle da Internet (ICMP – Internet Control Message Protocol) Um protocolo da pilha de protocolos TCP/IP que

lida com mensagens de erro e controle. Duas versões são usadas hoje, o ICMPv4 e o ICMPv6.

Protocolo de Reserva de Recursos (RSVP – Resource Reservation Protocol) Um protocolo de sinalização para ajudar a criar um fluxo IP e fazer uma reserva de recursos para melhorar a QoS.

Protocolo de Resolução de Endereços (ARP – Address Resolution Protocol) No TCP/IP, é um protocolo para a obtenção do endereço da camada de enlace de um nó quando o endereço Internet é conhecido.

Protocolo de Resolução de Endereços Reverso (RARP – Reverse Address Resolution Protocol) Um protocolo da pilha de protocolos TCP/IP que permite a um *host* determinar o seu endereço Internet dado o seu endereço físico.

protocolo de Retransmissão Seletiva Um protocolo de controle de erros no qual apenas o quadro com erros é retransmitido.

Protocolo de Roteamento de Borda (BGP – Border Gateway Protocol) Um protocolo de roteamento baseado em vetor de caminhos e usado entre sistemas autônomos.

protocolo de roteamento entre sistemas autônomos Um protocolo para lidar com as transmissões entre sistemas autônomos.

Protocolo de Roteamento *Multicast* por Vetor de Distâncias (DVMRP – Distance Vector Multicast Routing Protocol) Um protocolo de roteamento baseado em vetor de distâncias com suporte a roteamento *multicast*.

protocolo de Segurança da Camada de Transporte (TLS – Transport Layer Security) Um protocolo de segurança da camada de transporte projetado para fornecer segurança na WWW. Uma versão do protocolo SSL criada pela IETF.

Protocolo de Transferência de Arquivos (FTP – File Transfer Protocol) Um protocolo da camada de aplicação do modelo TCP/IP que transfere arquivos entre dois locais.

Protocolo de Transferência de Hipertexto (HTTP – HyperText Transfer Protocol) Um serviço da camada de aplicação que permite recuperar um documento Web.

Protocolo de Transporte em Tempo Real (RTP – Real-time Transport Protocol) Um protocolo para tráfego em tempo real; usado juntamente com o UDP.

protocolo de Troca de Chaves da Internet (IKE – Internet Key Exchange) Um protocolo projetado para criar associações de segurança no IPSec.

protocolo Diffie-Hellman Um protocolo de estabelecimento de chaves que fornece uma chave de sessão única para duas partes comunicantes.

protocolo em camadas O conceito de usar um conjunto de protocolos para criar uma hierarquia de regras com o objetivo de tratar uma tarefa difícil.

protocolo Go-Back-*N* (Volte Atrás *N*) Um protocolo de controle de erros em que o quadro com erros e todos os quadros seguintes devem ser retransmitidos.

Protocolo Internet (IP – Internet Protocol) O protocolo de camada de rede usado na pilha de protocolos TCP/IP, resposável pela transmissão não orientada à conexão em redes de comutação de pacotes. As duas versões comumente usadas são IPv4 e IPv6.

Protocolo Internet versão 6 (IPv6) A sexta versão do Protocolo Internet.

Protocolo Internet, nova geração (IPng – Internet Protocol, next generation) Outro termo usado para a sexta versão do Protocolo Internet, o IPv6.

protocolo *Multicast* Aberto de Menor Rota Primeiro (MOSPF – Multicast Open Shortest Path First) Um protocolo de *multicast* que usa roteamento *multicast* baseado em estado de enlace para criar uma árvore de menor custo baseada na origem.

protocolo orientado a *bits* Um protocolo no qual o quadro de dados é interpretado como uma sequência de *bits*.

protocolo orientado a *bytes* Um protocolo no qual o campo de dados de um quadro é interpretado como uma sequência de *bytes* (caracteres).

protocolo orientado a caracteres *Ver* protocolo orientado a bytes.

Protocolo Ponto a Ponto (PPP – Point-to-Point Protocol) Um protocolo para transferência de dados através de uma linha serial.

Protocolo Simples de Gerenciamento de Rede (SNMP – Simple Network Management Protocol) O protocolo TCP/IP que especifica o processo de gerenciamento na Internet.

Protocolo Simples de Transferência de Correio (SMTP – Simple Mail Transfer Protocol) O protocolo TCP/IP que especifica o serviço de correio eletrônico na Internet.

Protocolo Simples O protocolo simples que usamos para mostrar um método de acesso sem controle de fluxo e de erros.

protocolo *Stop-and-Wait* (Pare e Espere) Um protocolo em que o emissor envia um quadro, interrompe o envio até que receba a confirmação do receptor e envia o próximo quadro.

Provedor de Serviços de Internet (ISP – Internet Service Provider) Uma empresa que fornece serviços de Internet.

***proxy* ARP** Uma técnica que cria um efeito de criação de sub-redes; um servidor envia respostas ARP aos pedidos de vários nós.

proxy firewall Um *firewall* que filtra uma mensagem com base nas informações disponíveis na própria mensagem (na camada de aplicação).

pseudocabeçalho Informações do cabeçalho IP usadas apenas para o cálculo da soma de verificação (*checksum*) nos pacotes UDP e TCP.

pseudoternária Uma variação da codificação AM na qual um *bit* 1 é codificado com a tensão zero e um *bit* 0 é codificado alternando-se a tensão entre positivo e negativo.

psicoacústica Psicoacústica é o estudo da percepção humana subjetiva do som. A codificação perceptual tira proveito das falhas no sistema auditivo humano.

quadro (*frame*) Um grupo de *bits* que representa um bloco de dados.

quadro bidirecional (quadro B) Um quadro MPEG que está relacionado com o Quadro I ou Quadro P anterior e seguinte.

quadro de sinalização (*beacon frame*) Na função de coordenação pontual (PCF – Point Coordination Function) do projeto 802.11, refere-se a um quadro que inicia o intervalo de repetição.

Quadro I No MPEG, um Quadro I (intraquadro) é um quadro independente que não está relacionado com qualquer outro quadro (nem com o quadro enviado previamente nem com o enviado posteriormente). Eles aparecem em intervalos regulares.

quadro predito (quadro P) Um quadro predito é relacionado com o quadro I ou quadro B anterior. Em outras palavras, cada quadro P contém apenas as alterações do quadro anterior.

Qualidade de Serviço (QoS – Quality of Service) Um assunto relacionado a um conjunto de técnicas e mecanismos que garantem o desempenho de uma rede.

quantização A atribuição de uma faixa específica de valores para as amplitudes de um sinal.

quoted-printable Um esquema de codificação usado quando os dados consistem principalmente em caracteres ASCII com uma pequena parcela não ASCII.

radiação infravermelha Um tipo de radiação cuja frequência se encontra entre 300 GHz e 400 THz, geralmente usada para comunicações de curta distância.

Rádio Definido por *Software* (SDR – Software Defined Radio) Um sistema de comunicação de rádio em que os componentes de *hardware* tradicionais são implementados em *software*.

rajada de quadros (*frame bursting*) Uma técnica do Gigabit Ethernet com CSMA/CD em que múltiplos quadros são logicamente ligados uns aos outros para que fiquem semelhantes a um quadro mais longo.

rede A interconexão de um conjunto de dispositivos capazes de se comunicarem.

rede ad hoc Uma rede autoconfigurada conectada através de enlaces sem fios.

rede ANS (ANSNET – Advanced Networks and Services Network) A rede criada pela ANS.

Rede ARPA (ARPANET – Advanced Research Projects Agency Network) A rede de comutação de pacotes que foi criada pela ARPA. Foi usada para a pesquisa em interligação de redes.

Rede da Fundação Nacional de Ciência (NSFNET – National Science Foundation Network) A rede fundada pela Fundação Nacional de Ciência (NSF – National Science Foundation).

rede de células Uma rede que usa células como sua unidade básica de dados.

Rede de Ciência da Computação (CSNET – Computer Science Network) Uma rede patrocinada pela Fundação Nacional de Ciência (NSF – National Science Foundation), originalmente prevista para uso por universidades.

rede de comutação de circuitos Uma rede na qual a tecnologia de comutação de circuitos é utilizada. Um bom exemplo é a antiga rede de telefonia de voz.

rede de comutação de pacotes Uma rede na qual os dados são transmitidos em unidades independentes chamadas pacotes.

rede de confiança (*Web of trust*) No PGP, refere-se aos molhos de chave compartilhados por um grupo de pessoas.

rede de datagramas Uma rede de comutação de pacotes na qual os pacotes são independentes uns dos outros.

Rede de Longa Distância (WAN – Wide Area Network) Uma rede que usa uma tecnologia que pode cobrir uma grande distância geográfica.

rede de satélite Uma combinação de nós que proporciona comunicação de um ponto da Terra até outro.

rede de telefonia convencional (POTS – plain old telephone system) A rede de telefonia convencional usada para comunicação por voz.

rede de TV a cabo Um sistema que usa cabos coaxiais ou de fibra ótica para trazer múltiplos canais de vídeo para as residências.

rede externa (rede visitada, ou *foreign network*) A rede à qual um nó móvel está conectado, diferente da sua rede original.

rede híbrida fibra/coaxial (HFC – hybrid-fiber-coaxial) A segunda geração de redes com fios; utiliza cabos de fibra óptica e coaxiais.

rede híbrida Uma rede contendo uma internet privada e acesso à Internet global.

rede local (LAN – local area network) Uma rede que conecta dispositivos dentro de uma única edificação ou no interior de edificações próximas umas às outras.

Rede Local Virtual (VLAN – Virtual Local Area Network) Uma tecnologia que divide uma LAN física em grupos de trabalho virtuais usando *software*.

Rede Metropolitana (MAN – Metropolitan Area Network) Uma rede que pode abranger uma área geográfica do tamanho de uma cidade.

Rede Militar (MILNET – Military Network) Uma rede para uso militar que era originalmente parte da ARPANET.

rede nativa (*home network*) Uma rede que é a morada permanente do nó móvel.

Rede Óptica Síncrona (SONET – Synchronous Optical Network) Um padrão desenvolvido pela ANSI para tecnologia de fibra óptica que pode transmitir dados em alta velocidade. Tal tecnologia pode ser utilizada para transmitir texto, áudio e vídeo.

rede privada Uma rede que está isolada da Internet.

Rede Privada Virtual (VPN – Virtual Private Network) Uma tecnologia que cria uma rede que é fisicamente pública, mas virtualmente privada.

redução multiplicativa Uma técnica para evitar o congestionamento em que o limiar é definido como a metade do tamanho da última janela de congestionamento e o tamanho da janela de congestionamento é reiniciado em um.

redundância Adição de *bits* a uma mensagem para auxiliar no controle de erros e tratamento deles.

Reed-Solomon Um código cíclico complexo, mas eficiente.

reflexão Fenômeno pelo qual a luz é refletida na fronteira entre dois meios.

refração Fenômeno pelo qual a luz se desvia de sua trajetória original quando passa de um meio para outro.

registrador de deslocamento Um registrador em que cada localização de memória, em certo pulso do relógio, recebe o *bit* de sua porta de entrada, armazena o novo *bit* e exibe-o na porta de saída.

registradores de domínios (*registrar*) Uma autoridade para registrar novos nomes de domínio.

Regras Básicas de Codificação (BER – Basic Encoding Rules) Um padrão para codificação de dados que são transferidos por uma rede.

Relação Sinal/Ruído (SNR – Signal-to-Noise Ratio) A razão entre a potência média do sinal e a média da potência do ruído.

repetidor Um dispositivo que expande a distância que um sinal pode viajar através da regeneração do sinal.

representação no domínio do tempo Uma representação gráfica da amplitude de um sinal no tempo.

resolução de nomes/endereços Mapeamento de um nome para um endereço ou vice-versa.

resolução iterativa Resolução de endereços IP na qual o cliente pode enviar o seu pedido para vários servidores antes de obter uma resposta.

resolução recursiva Processo de resolução de endereços IP no qual o cliente envia seu pedido a um servidor que, por fim, retorna uma resposta.

resolvedor O cliente DNS que é usado por uma máquina que precisa mapear um endereço para um nome ou um nome para um endereço.

resumo criptográfico (*digest*) Uma versão condensada de um documento.

Resumo Criptográfico da Mensagem (MD – Message Digest) Um conjunto de vários algoritmos de *hash* projetados por Ron Rivest, denominados: MD2, MD4 e MD5.

Retorno ao Zero (RZ) Uma técnica de codificação digital-digital em que a tensão do sinal é zero para a segunda metade do intervalo de *bits*.

retransmissão rápida (*fast retransmission*) Retransmissão de um segmento no protocolo TCP após três ACKs duplicados serem recebidos, o que indica que aquele segmento foi perdido ou corrompido.

revestimento (*cladding*) Revestimento de vidro ou de plástico que envolve o núcleo de uma fibra óptica; a densidade óptica desse revestimento deve ser menor do que a do núcleo.

Rivest, Shamir, Adleman (RSA) *Ver* sistema criptográfico RSA.

RJ45 Um conector para cabo de par trançado.

roaming No contexto de telefonia celular, refere-se à capacidade de um usuário de se comunicar fora da área do seu provedor de serviços. Também denominado itinerância.

rodapé (*trailer*) Informação de controle anexada ao final da unidade de dados.

rota específica Uma entrada na tabela de roteamento contendo o endereço IP completo de um nó.

rota Um caminho percorrido por um pacote.

roteador de encontro Um roteador que é o núcleo ou centro para cada grupo de *multicast*; ele se torna a raiz da árvore.

roteador *multicast* Um roteador que relaciona membros interessados em certo conteúdo a cada interface do roteador, distribuindo os pacotes *multicast* de acordo.

roteador Um dispositivo para interligação de redes que opera nas três primeiras camadas da pilha de protocolos TCP/IP. Um roteador é ligado a duas ou mais redes e encaminha pacotes de uma rede para outra.

roteamento baseado em estado de enlace Um método de roteamento em que cada roteador compartilha seu conhecimento sobre as alterações na sua vizinhança com todos os outros roteadores.

roteamento baseado em vetor de distâncias Um método de roteamento no qual cada roteador envia a seus vizinhos uma lista de redes que ele pode alcançar e a distância a cada uma dessas redes.

roteamento baseado na origem (*source routing*) Ato de definir explicitamente o percurso de um pacote com base em sua origem.

roteamento baseado no próximo salto (*next-hop*) Um método de roteamento no qual apenas o endereço do próximo salto está listado na tabela de roteamento e não uma lista completa dos saltos pelos quais o pacote deve passar.

roteamento de rede específica Roteamento no qual todos os *hosts* em uma rede compartilham uma entrada na tabela de roteamento.

roteamento dinâmico Roteamento no qual as entradas da tabela são atualizadas automaticamente pelo protocolo de roteamento.

roteamento estático Um tipo de roteamento no qual a tabela de roteamento permanece inalterada.

roteamento externo Roteamento entre sistemas autônomos.

roteamento geográfico Uma técnica de roteamento na qual o espaço de endereços inteiro é dividido em blocos com base em localizações geográficas.

roteamento hierárquico Uma técnica de roteamento na qual o espaço de endereços inteiro é dividido em níveis, com base em critérios específicos.

roteamento interdomínios Refere-se ao roteamento entre sistemas autônomos.

Roteamento Interdomínios sem Classes (CIDR – Classless Interdomain Routing) Uma técnica para fazer roteamento usando endereçamento sem classes.

roteamento interno Roteamento que ocorre no interior de um sistema autônomo.

roteamento *multicast* O ato de encaminhar um pacote *multicast* para os seus destinos.

roteamento O processo, realizado por um roteador, de determinar o próximo salto para um datagrama.

roteamento padrão Um método de roteamento segundo o qual um roteador é designado para receber todos os pacotes para os quais não haja correspondência na sua tabela de roteamento.

roteamento por vetor de distâncias Um método de roteamento no qual o BGP se baseia; nesse método, os ASs através do qual um pacote deve passar são explicitamente listados.

roteamento triangular No IP móvel, o caso menos grave de ineficiência que ocorre quando o *host* remoto se comunica com um *host* móvel que não está ligado à mesma rede (ou local) que o *host* móvel.

rótulo (*label*) Um identificador usado em serviços orientados à conexão para definir o caminho (isto é, a localização do serviço).

ruído pseudoaleatório (RP) Um gerador de código pseudoaleatório usado no FHSS.

ruído Sinais elétricos aleatórios que podem ser captados no meio de transmissão e que causam uma degradação ou distorção dos dados.

scatternet Uma combinação de piconets.

segmentação A divisão de uma mensagem em vários pacotes, geralmente realizada na camada de transporte.

Segmentação e Remontagem (SAR – Segmentation and Reassembly) A parte mais baixa da subcamada AAL no protocolo ATM, na qual um cabeçalho e/ou rodapé (*trailer*) pode ser adicionado para produzir um elemento de 48 *bytes*.

segmento O pacote da camada TCP. Também se refere ao comprimento do meio de transmissão compartilhado por dispositivos.

segredo-mestre No SSL, um segredo de 48 *bytes* criado a partir do *segredo pré-mestre*.

segredo pré-mestre No SSL, um segredo trocado entre o cliente e o servidor antes do cálculo do segredo mestre.

seleção No método de acesso por varredura/seleção, refere-se a um procedimento no qual a estação primária pergunta a uma estação secundária se ela está pronta para receber dados.

semifechada (*half-close*) No TCP, refere-se a um tipo de encerramento de conexão no qual um lado interrompe o envio de dados enquanto ainda os está recebendo.

sequência de Barker Uma sequência de 11 *bits* usada para espalhamento espectral.

sequência ortogonal Uma sequência com propriedades especiais entre os elementos.

serviço de múltiplos fluxos Um serviço fornecido pelo SCTP que permite que a transferência de dados seja realizada usando diferentes fluxos.

serviço de Sinal Digital (DS – Digital Signal) Um serviço provido por companhias telefônicas que conta com uma hierarquia de sinais digitais.

Serviço Digital de Dados (DDS – Digital Data Service) Uma versão digital de uma linha analógica dedicada com uma taxa de dados de 64 Kbps.

serviço multiprovedor Um serviço fornecido pelo protocolo SCTP no qual um *host* pode ser ligado a mais de uma rede.

serviço orientado à conexão Um serviço de transferência de dados envolvendo o estabelecimento e o encerramento de uma conexão.

serviço sem conexão Um serviço de transferência de dados que não envolve estabelecimento e encerramento de uma conexão.

Serviços & Redes Avançados (ANS – Advanced Network and Services) Uma organização sem fins lucrativos criada pela IBM, Merit e MCI para construir um *backbone* de alta velocidade.

Serviços Diferenciados (DS ou DiffServ – Differentiated Services) Um modelo de QoS baseado em classes de tráfego projetado para uso no protocolo IP.

Serviços Integrados (IntServ – Integrated Services) Um modelo de QoS baseado em fluxos, projetado para uso no protocolo IP.

servidor concorrente orientado à conexão Um servidor orientado à conexão que pode servir diversos clientes simultaneamente.

servidor concorrente Um servidor que serve vários clientes simultaneamente.

servidor de mídia Um servidor usado para transmitir fluxos de áudio ou vídeo.

servidor de registro de domínio No SIP, um servidor com o qual um usuário é registrado a cada instante.

servidor DNS Um computador que tem informações sobre o espaço de nomes (*namespace*).

servidor iterativo sem conexão Um servidor sem conexão que processa uma requisição de cada vez.

servidor iterativo Um servidor que pode prover serviços a apenas um cliente de cada vez.

servidor *proxy* Um computador que mantém cópias das respostas dos pedidos mais recentes.

servidor raiz No DNS, refere-se a um servidor cuja zona corresponde à árvore inteira. Um servidor-raiz normalmente não armazena qualquer informação sobre domínios, mas delega a sua autoridade a outros servidores, mantendo as referências a eles.

servidor Um programa que pode prestar serviços a outros programas, chamados de clientes.

Setor de Padronização da ITU (ITU-T – ITU Standardization Sector) A organização de padronização anteriormente conhecida como CCITT.

Shell Seguro (SSH – Secure Shell) Um programa cliente-servidor que fornece acesso remoto seguro.

sinal analógico Uma forma de onda contínua que varia suavemente ao longo do tempo.

sinal aperiódico Um sinal que não exibe um padrão ou ciclo de repetição.

sinal aperiódico Um sinal que não tem qualquer periodicidade; que não exibe um padrão ou ciclo de repetição.

sinal composto Um sinal composto por uma ou mais ondas senoidais.

sinal de interferência (*jamming signal*) No CSMA/CD, refere-se a um sinal enviado pela primeira estação que detecta colisão para alertar todas as outras estações dessa situação.

sinal de portadora Um sinal de alta frequência usado em modulação digital-analógica ou analógica-analógica. Uma das características do sinal de portadora (amplitude, frequência ou fase) é alterada de acordo com os dados sendo modulados.

Sinal de Transporte Síncrono (STS) Um sinal na hierarquia SONET.

sinal digital Um sinal discreto com um número limitado de valores.

sinal periódico Um sinal que apresenta um padrão de repetição.

sinalização em banda (*inband*) Usa o mesmo canal tanto para dados como para tráfego de controle.

sinalização fora da banda O uso de dois canais separados para enviar dados e controle.

síndrome da janela boba (*silly window syndrome*) Uma situação em que uma janela de tamanho pequeno é anunciada pelo receptor e um segmento pequeno é enviado pelo emissor.

síndrome Uma sequência de *bits* gerados por meio da aplicação da função de verificação de erros a uma palavra de código.

Sintaxe para Mensagens Cifradas (CMS – Cryptographic Message Syntax) A sintaxe usada no S/MIME que define exatamente o esquema de codificação para cada tipo de conteúdo.

Sistema Autônomo (AS – Autonomous System) Um grupo de redes e roteadores sob a autoridade de uma mesma entidade administrativa.

Sistema Avançado de Telefonia Móvel (AMPS – Advanced Mobile Phone System) Um sistema norte-americano de telefonia celular analógica que utiliza FDMA.

sistema criptográfico RSA Um método popular de criptografia de chave pública, desenvolvido por Rivest, Shamir e Adleman.

Sistema de Antenas Adaptativas (AAS – Adaptive Antenna System) Um sistema que utiliza múltiplas antenas tanto no terminal como na estação base para aumentar o desempenho.

Sistema de Comunicação Pessoal (PCS – Personal Communication System) Um termo genérico para um sistema comercial de celular que fornece vários tipos de serviços de comunicação.

Sistema de Nomes de Domínio (DNS – Domain Name System) Um serviço TCP/IP da camada de aplicação que converte "nomes amigáveis" em endereços IP.

Sistema de Nomes de Domínio Dinâmico (DDNS – Dynamic Domain Name System) Um método para atualizar dinamicamente o arquivo mestre do DNS.

Sistema de Posicionamento Global (GPS – Global Positioning System) Um sistema de satélites público composto por 24 satélites MEO e usado para navegação terrestre e marítima. O GPS não é usado para comunicações.

Sistema de Telecomunicações Móveis Universal (UMTS – Universal Mobile Telecommunication System) Uma das tecnologias 3G populares usando uma versão do CDMA denominada CDMA de Banda Larga com Sequência Direta (DS-WCDMA – Direct Sequence Wideband CDMA).

sistema de transmissão por modem a cabo (CMTS – cable modem transmission system) Um dispositivo instalado dentro da central de distribuição do provedor de TV a cabo, recebendo dados da Internet e passando-os para o circuito combinador.

sistema final Um emissor ou receptor de dados.

Sistema Global para Comunicações Móveis (GSM – Global System for Mobile Communication) Um sistema de telefonia celular da segunda geração bastante usado na Europa e no Brasil.

Sociedade da Internet (ISOC – Internet Society) A organização sem fins lucrativos criada para promover o uso da Internet.

***socket* de fluxo** Uma estrutura projetada para ser usada com um protocolo orientado à conexão como o TCP.

***socket* puro (*raw socket*)** Uma estrutura projetada para protocolos que utilizam diretamente os serviços de IP e não utilizam *sockets* de fluxo nem *sockets* de datagrama.

socket Um ponto final para um processo; dois *sockets* são necessários para uma comunicação.

Solução de Clark Uma solução para evitar a "síndrome da janela boba" (*silly window syndrome*). Uma confirmação é enviada assim que os dados chegam, porém ela anuncia um tamanho de janela igual a zero até que haja espaço suficiente para acomodar um segmento de tamanho máximo ou até que metade do *buffer* esteja vazio.

soma de verificação (*checksum*) Um valor utilizado para a detecção de erros. Ele é calculado somando-se as unidades de dados usando aritmética em complemento de um e então complementando-se o resultado.

STREAM Uma das interfaces que foram definidas para programação de redes.

subcamada de Controle de Acesso ao Meio (MAC – Media Access Control) A subcamada mais baixa da camada de enlace de dados definida pelo projeto IEEE 802. Ela define o método de acesso e controle de acesso em diferentes protocolos de rede local.

Subcamada de Convergência (CS – Converge Sublayer) No protocolo ATM, refere-se à subcamada AAL mais alta, que acrescenta um cabeçalho e um rodapé aos dados do usuário.

subcamada PHY O transceptor na Fast Ethernet.

subnet Uma sub-rede.

sub-rede Uma parte de uma rede.

substituição monoalfabética Um método de criptografia no qual cada ocorrência de um caractere é substituída por outro caractere do conjunto.

substituição polialfabética Um método de criptografia no qual cada ocorrência de um caractere pode ter um substituto diferente.

sufixo A parte variável (semelhante ao *hostid*) de um endereço IP.

supergrupo Um sinal composto por cinco grupos multiplexados.

super-rede Uma rede formada a partir de duas ou mais redes de dimensões menores.

***switch* de armazenamento e encaminhamento (*store-and-forward*)** Um *switch* que armazena o quadro em um *buffer* de entrada até o pacote inteiro chegar.

switch Um dispositivo conectando múltiplas linhas de comunicação.

Tabela de *Hash* Distribuída (DHT – Distributed Hash Table) Uma DHT distribui os dados (ou referências aos dados) entre um conjunto de nós de acordo com algumas regras predefinidas. Cada nó em uma rede baseada em DHT torna-se responsável por uma faixa desses dados.

tabela de roteamento Uma tabela contendo as informações de que um roteador precisa para rotear pacotes. As informações podem incluir o endereço de rede, custos, o endereço do próximo salto e assim por diante.

tabela de Walsh No CDMA, refere-se a uma tabela bidimensional usada para gerar sequências ortogonais.

tamanho da rajada autorizado (Bc – committed burst size) O número máximo de *bits* em um determinado período de tempo que uma rede deve transmitir sem descartar os quadros.

Taxa Constante de *Bits* (CBR – Constant Bit Rate) A taxa de transmissão de dados de uma classe de serviço ATM, projetada para clientes que necessitem de serviços de tempo real para áudio ou vídeo.

taxa de amostragem O número de amostras obtidas por segundo no processo de amostragem.

taxa de *bits* (*bit rate*) O número de *bits* transmitidos por segundo.

Taxa de *Bits* de Nyquist A taxa de transferência de dados com base no teorema de Nyquist.

Taxa de *Bits* Não Especificada (UBR – Unspecified Bit Rate) A taxa de dados de uma classe de serviços

ATM que especifica o uso apenas da entrega de melhor esforço.

Taxa de *Bits* Variável (VBR – Variable Bit Rate) A taxa de dados de uma classe de serviços ATM para usuários que necessitam de uma taxa de *bits* variável.

taxa de dados O número de elementos de dados enviados por segundo.

taxa de informações autorizada (CIR – committed information rate) O tamanho da rajada consignado dividido pelo tempo.

taxa de transferência do sinal O número de elementos de sinal enviados em um segundo.

taxa de transmissão O número de *bits* enviados por segundo.

TDM estatística Uma técnica TDM na qual parcelas discretas (*slots*) são alocadas dinamicamente para melhorar a eficiência.

TDM síncrona Uma técnica de TDM na qual cada entrada tem um espaço alocado na saída, mesmo quando não estiver enviando dados.

TDMA com Duplexação por Divisão de Tempo (TDD--TDMA – Time-Division Duplex TDMA) Em uma rede Bluetooth, um tipo de comunicação *half--duplex* na qual o escravo e o receptor enviam e recebem dados, mas não ao mesmo tempo (*half-duplex*).

Técnica de Múltiplos Tons Discretos (DMT – Discrete Multitone Technique) Um método de modulação que combina elementos de QAM e FDM.

telecomunicações Troca de informações à distância utilizando equipamentos eletrônicos.

teleconferência Comunicação de áudio e vídeo entre usuários remotos.

Teledesic Um sistema de satélites que fornece comunicação de fibra óptica (canais de banda larga, baixa taxa de erros e reduzido atraso).

telefonia celular Uma técnica de comunicação sem fio na qual uma área é dividida em células. Cada célula é servida por um transmissor.

Tempo de Ida e Volta (RTT – Round-Trip Time) O tempo necessário para um datagrama ir da sua origem até o destino e voltar.

tempo de propagação O tempo necessário para um sinal se deslocar de um ponto para outro.

Tempo de Vida (TTL – Time-To-Live) O tempo de vida de um pacote.

Tempo-Limite de Retransmissão (RTO – Retransmission Time-Out) A expiração de um temporizador que controla a retransmissão dos pacotes.

temporizador de manutenção de sessão (*keepalive*) Um temporizador do protocolo TCP que verifica se há um processo ativo em outra estação.

temporizador de persistência Um temporizador do TCP utilizado para evitar impasse (*deadlock*).

Teorema de Nyquist Um teorema que indica que o número de amostras necessárias para representar adequadamente um sinal analógico é igual a duas vezes a frequência mais alta do sinal original.

Terminal de Rede (TELNET – Terminal Network) Um programa cliente-servidor de propósito geral para acesso remoto.

Terminal Virtual de Rede (NVT – Network Virtual Terminal) Um protocolo de aplicação TCP/IP que permite acesso remoto.

texto às claras (*plaintext*) A mensagem antes de ser cifrada ou após ser decifrada.

texto cifrado A mensagem após ser cifrada.

tipo de dado estruturado Um tipo de dados complexo constituído por alguns tipos simples ou dados estruturados.

Tipo de Serviço (TOS – Type of Service) Um critério ou valor que especifica o tratamento a ser dado ao datagrama.

topologia A estrutura de uma rede incluindo a disposição física dos dispositivos.

topologia em anel Uma topologia na qual os dispositivos estão ligados em anel. Cada dispositivo no anel recebe a unidade de dados proveniente do dispositivo anterior, regenera-o e o encaminha para o dispositivo seguinte.

topologia em estrela Uma topologia na qual todas as estações estão ligadas a um dispositivo central (*hub*).

topologia em malha (*mesh*) Uma configuração de rede na qual cada dispositivo apresenta uma ligação ponto a ponto dedicada para todos os outros dispositivos.

Tradução de Endereços de Rede (NAT – Network Address Translation) Uma tecnologia que permite que uma rede privada possa usar um conjunto de endereços privados para comunicação interna e um conjunto de endereços globais da Internet para comunicação externa.

tráfego multimídia Tráfego constituído por dados, vídeo e áudio.

transceptor Um dispositivo que tanto transmite como recebe.

transferência (*handoff*) Evento de mudança para um novo canal quando um dispositivo móvel se movimenta de uma célula para outra.

Transformada Discreta do Cosseno (DCT – Discrete Cosine Transform) Uma técnica de compressão na qual o sinal representando dados é transformado do domínio tempo-espaço para o domínio da frequência.

transmissão em banda base Transmissão de um sinal digital ou analógico sem modulação, utilizando um canal passa-baixas.

transmissão em banda larga Transmissão de sinais utilizando modulação de um sinal de frequência mais alta. O termo se refere a dados combinados de diferentes fontes que ocupam uma ampla largura de banda.

transmissão paralela Transmissão na qual *bits* agrupados são enviados simultaneamente, cada um usando um enlace separado.

transmissão serial Transmissão de dados um *bit* por vez, usando um único enlace.

transmissão síncrona Um método de transmissão que requer uma relação de tempo constante entre o emissor e o receptor.

transmissão *unicast* O envio de um pacote para apenas um destino.

transmissão via *broadcast* Transmissão de uma mensagem para todos os nós de uma rede.

transparência A capacidade de enviar qualquer sequência de *bits* como dados sem que eles sejam confundidos com *bits* de controle.

transparência de dados A capacidade de enviar qualquer padrão de *bits* como dados sem que eles sejam confundidos com *bits* de controle.

travessia dupla (*double crossing*) No contexto de IP móvel, dá-se o nome de travessia dupla quando um nó remoto se comunica com um nó móvel que se moveu para a mesma rede que o nó remoto.

triangulação O mesmo que trilateração, mas utilizando três ângulos, em vez de três distâncias.

Tributários Virtuais (VT – Virtual Tributary) Uma carga útil (*payload*) parcial que pode ser inserida em um quadro SONET e combinada com outras cargas úteis parciais para preencher o quadro.

trilateração Um método bidimensional de determinar uma localização dadas as distâncias a três pontos diferentes.

trunk meios de transmissão que lidam com a comunicação entre estações.

túnel lógico O encapsulamento de um pacote *multicast* dentro de um pacote *unicast* para permitir o roteamento *multicast* por roteadores sem suporte a *multicast*.

tunelamento No *multicast*, um processo no qual o pacote *multicast* é encapsulado em um pacote *unicast* e enviado pela rede. Em uma VPN, refere-se ao encapsulamento de um datagrama IP cifrado em um segundo datagrama de saída. No IPv6, refere-se a uma estratégia usada quando dois computadores que usam IPv6 desejam se comunicar um com o outro, mas o pacote deve passar por uma região que usa IPv4.

União Internacional de Telecomunicações (ITU – International Telecommunications Union) Uma organização internacional de telecomunicações.

***unicast* múltiplo** O envio de múltiplas cópias de uma mensagem, cada uma com um endereço *unicast* de destino diferente, a partir de uma fonte.

Unicode O conjunto internacional de caracteres usado para definir os caracteres válidos em ciência da computação.

Unidade Máxima de Transferência (MTU – Maximum Transfer Unit) O maior tamanho da unidade de dados que uma rede específica pode tratar.

upload Envio de um arquivo ou dados locais para um local remoto.

variação da linha de base (*baseline wandering*) Na decodificação de um sinal digital, o receptor calcula uma média móvel da potência do sinal recebido. Essa média é chamada "linha de base". Uma sequência longa de 0s ou 1s pode causar um desvio na linha de base (*baseline wandering*), e dificultar a decodificação correta pelo receptor.

variação Em uma rede HFC, um processo que determina a distância entre o CM e o CMTS.

varredura (*polling*) Um método de acesso no qual um dispositivo é designado como uma estação primária e os outros como as estações secundárias. O acesso é controlado pela estação primária.

varredura/seleção Um protocolo de método de acesso utilizado nos procedimentos de varredura e seleção. *Ver* varredura. *Ver* seleção.

varrer No método de acesso primário/secundário, um procedimento pelo qual a estação primária pergunta a cada estação secundária se ela tem algum dado para transmitir.

vazão O número de *bits* que podem passar de um ponto a outro.

velocidade de propagação A rapidez na qual um sinal ou *bit* viaja; medida em distância/segundo.

verificação de paridade em duas dimensões Um método de detecção de erros bidimensional.

Verificação de Redundância Cíclica (CRC – Cyclic Redundancy Check) Um método de detecção de erros de alta precisão, no qual uma sequência de *bits* é interpretada como um polinômio.

Vetor de Alocação de Rede (NAV – Network Allocation Vector) No CSMA/CA, a quantidade de tempo que deve se passar antes que uma estação possa verificar se o meio está ocioso.

vídeo A gravação ou a transmissão de uma imagem ou um filme.

voz sobre IP Uma tecnologia na qual a Internet é usada como uma rede telefônica.

World Wide Web (WWW) Um serviço multimídia da Internet que permite aos usuários navegar na Internet, passando de um documento para outro através de links que os conectam.

X.509 Uma recomendação elaborada pela ITU e aceita pela Internet que define os certificados de uma forma estruturada.

zona No DNS, refere-se á região que é de responsabilidade de um servidor ou sobre a qual ele tem autoridade.

REFERÊNCIAS

[AL 98] Albitz, P., and Liu, C. *DNS and BIND,* 3rd ed. Sebastopol, CA: O'Reilly, 1998.

[AZ 03] Agrawal, D., and Zeng, Q. *Introduction to Wireless and Mobile Systems.* Pacific Grove, CA: Brooks/Cole Thomson Learning, 2003.

[Bar *et al.* 05] Barrett, Daniel J., Silverman, Ricard E., and Byrnes, Robert G. *SSH: The Secure Shell: The Definitive Guide,* Sebastopol, CA: O'Reilly, 2005.

[BEL 01] Bellamy, J. *Digital Telephony.* New York, NY: Wiley, 2001.

[Ber 96] Bergman, J. *Digital Baseband Transmission and Recording.* Boston, MA: Kluwer, 1996.

[Bis 03] Bishop, D. *Introduction to Cryptography with Java Applets.* Sebastopol, CA: O'Reilly, 2003.

[Bis 05] Bishop, Matt. *Introduction to Computer Security.* Reading, MA: Addison-Wesley, 2005.

[Bla 00] Black, U. *QOS in Wide Area Networks.* Upper Saddle River, NJ: Prentice Hall, 2000.

[Bla 00] Black, U. *PPP and L2TP: Remote Access Communication.* Upper Saddle River, NJ: Prentice Hall, 2000.

[BYL 09] Buford, J. F., Yu, H., and Lua, E. K. *P2P Networking and Applications.* San Francisco: Morgan Kaufmann, 2009.

[CD 08] Calvert, Kenneth L., and Donaho, Michael J. *TCP/IP Sockets in Java.* San Francisco, CA: Morgan Kaufmann, 2008.

[CHW 99] Crowcroft, J., Handley, M., Wakeman, I. *Internetworking Multimedia.* San Francisco, CA: Morgan Kaufmann, 1999.

[Com 06] Comer, Douglas E. *Internetworking with TCP/IP,* vol. 1. Upper Saddle River, NJ: Prentice Hall, 2006.

[Cou 01] Couch, L. *Digital and Analog Communication Systems.* Upper Saddle River, NJ: Prentice Hall, 2001.

[DC 01] Donaho, Michael J., and Calvert, Kenneth L. *TCP/IP Sockets: C version.* San Francisco, CA: Morgan Kaufmann, 2001.

[DH 03] Doraswamy, H., and Harkins, D. *IPSec.* Upper Saddle River, NJ: Prentice Hall, 2003.

[Dro 02] Drozdek A. *Elements of Data Compression.* Pacific Grove, CA: Brooks/Cole (Thomson Learning), 2002.

[Far 04] Farrel, A. *The Internet and Its Protocols.* San Francisco: Morgan Kaufmann, 2004.

[FH 98] Ferguson, P., and Huston, G. *Quality of Service.* New York: John Wiley and Sons, Inc., 1998.

[For 03] Forouzan, B. *Local Area Networks.* New York: McGraw-Hill, 2003.

[For 07] Forouzan, B. *Introduction to Data Communication and Networking.* New York: McGraw-Hill, 2007.

[For 08] Forouzan, B., *Cryptography and Network Security.* New York: McGraw-Hill, 2008.

[For 08] Forouzan, B., *TCP/IP Protocol Suite.* New York: McGraw-Hill, 2010.

[Fra 01] Frankkel, S. *Demystifying the IPSec Puzzle.* Norwood, MA: Artech House, 2001.

[GW 04] Garcia, A., and Widjaja, I. *Communication Networks.* New York, NY: McGraw-Hill, 2004.

[Gar 01] Garret, P. *Making, Breaking Codes.* Upper Saddle River, NJ: Prentice Hall, 2001.

[Gar 95] Garfinkel, S. *PGP: Pretty Good Privacy.* Sebastopol. CA: O'Reilly, 1995.

[Gas 02] Gast, M. *802.11 Wireless Networks.* Sebastopol, CA: O'Reilly, 2002.

[GGLLB 98] Gibson, J. D., Gerger, T., Lookabaugh, T., LindBerg, D., and Baker, R. L. *Digital Compression for Multimedia.* San Francisco: Morgan Kaufmann, 1998.

|Hal 01| Halsall, F. *Multimedia Communication*. Reading, MA: Addison- Wesley, 2001.

|Ham 80| Hamming, R. *Coding and Information Theory*. Upper Saddle River, NJ: Prentice Hall, 1980.

|Har 05| Harol, Elliot R. *Java Network Programming*. Sebastopol, CA: O'Reilly, 2005.

|HM 10| Havaldar, P., and Medioni, G. *Multimedia Systems: Algorithms, Stan-dards, and Industry Practices*. Boston: Course Technology (Cengage Learning), 2010.

|Hsu 03| Hsu, H. *Analog and Digital Communications*. New York, NY: McGraw-Hill, 2003.

|Hui 00| Huitema, C. *Routing in the Internet*, 2nd ed. Upper Saddle River, NJ: Prentice Hall, 2000.

|Izz 00| Izzo, P. *Gigabit Networks*. New York, NY: Wiley, 2000.

|Jam 03| Jamalipour, A. *Wireless Mobile Internet*. New York, NY: Wiley, 2003.

|Jen *et al.* 86| Jennings, D. M., Landweber, L. M., Fuchs, I. H., Farber, D. H., and Adrion, W. R. "Computer Networking for Scientists and Engineers," *Science* 231, no. 4741 (1986): 943–950.

|KCK 98| Kadambi, J., Crayford, I., and Kalkunte, M. *Gigabit Ethernet*. Upper Saddle River, NJ: Prentice Hall, 1998.

|Kei 02| Keiser, G. *Local Area Networks*. New York, NY: McGraw-Hill, 2002.

|Kes 02| Keshav, S. *An Engineering Approach to Computer Networking*. Reading, MA: Addison-Wesley, 2002.

|Kle 04| Kleinrock, L. *The Birth of the Internet*.

|KMK 04| Kumar, A., Manjunath, D., and Kuri, J. *Communication Network: An Analytical Approach*. San Francisco: Morgan Kaufmann, 2004.

|Koz 05| Kozierock, Charles M. *The TCP/IP Guide*. San Francisco: No Starch Press, 2005.

|Lei *et al.* 98| Leiner, B., Cerf, V., Clark, D., Kahn, R., Kleinrock, L., Lynch, D., Postel, J., Roberts, L., and Wolff, S., *A Brief History of the Internet*. http://www.isoc.org/internet/history/brief.shtml.

|Los 04| Loshin, Pete. *IPv6: Theory, Protocol, and Practice*. San Francisco: Morgan Kaufmann, 2004.

|Mao 04| Mao, W. *Modern Cryptography*. Upper Saddle River, NJ: Prentice Hall, 2004.

|Max 99| Maxwell, K. *Residential Broadband*. New York, NY: Wiley, 1999.

|Mir 07| Mir, Nader F. *Computer and Communication Networks*. Upper Saddle River, NJ: Prentice Hall, 2007.

|Moy 98| Moy, John. *OSPF*. Reading, MA: Addison-Wesley, 1998.

|MS 01| Mauro, D., and Schmidt, K. *Essential SNMP*. Sebastopol, CA: O'Reilly, 2001.

|PD 03| Peterson, Larry L., and Davie, Bruce S. *Computer Networks*, 3rd ed. San Francisco: Morgan Kaufmann, 2003.

|Pea 92| Pearson, J. *Basic Communication Theory*. Upper Saddle River, NJ: Prentice Hall, 1992.

|Per 00| Perlman, Radia. *Interconnections*, 2nd ed. Reading, MA: Addison-Wesley, 2000.

|Pit 06| Pitt, E. *Fundamental Networking in Java*. Berlin: Springer-Verlag, 2006.

|PKA 08| Poo, D., Kiong, D., Ashok, S. *Object-Oriented Programming and Java*. Berlin: Springer-Verlag, 2008.

|Res 01| Rescorla, E. *SSL and TLS*. Reading, MA: Addison-Wesley, 2001.

|Ror 96| Rorabaugh, C. *Error Coding Cookbook*. New York, NY: McGraw-Hill, 1996.

|RR 96| Robbins, Kay A., and Robbins, Steven. *Practical UNIX Program- ming*, Upper Saddle River, NJ: Prentice Hall, 1996.

|Sau 98| Sauders, S. *Gigabit Ethernet Handbook*. New York, NY: McGraw-Hill, 1998.

|Sch 03| Schiller, J. *Mobile Communications*. Reading, MA: Addison- Wesley, 2003.

|Seg 98| Segaller, S. Nerds 2.0.1: *A Brief History of the Internet*. New York: TV Books, 1998.

|Sna 00| Snader, J. C. *Effective TCP/IP Programming*. Reading, MA: Addison-Wesley, 2000.

|Spi 74| Spiegel, M. *Fourier Analysis*. New York, NY: McGraw-Hill, 1974.

|Spu 00| Spurgeon, C. *Ethernet*. Sebastopol, CA: O'Reilly, 2000.

|Sta 02| Stallings, W. *Wireless Communications and Networks*. Upper Saddle River, NJ: Prentice Hall, 2002

[Sta 04] Stallings, W. *Data and Computer Communications,* 7th ed. Upper Saddle River, NJ: Prentice Hall, 2004.

[Sta 06] Stallings, W. *Cryptography and Network Security,* 5th ed. Upper Saddle River, NJ: Prentice Hall, 2006.

[Ste 94] Stevens, W. Richard. *TCP/IP Illustrated,* vol. 1. Reading, MA: Addison-Wesley, 1994.

[Ste 95] Stevens, W. Richard. *TCP/IP Illustrated,* vol. 2. Reading, MA: Addison-Wesley, 1995.

[Ste 99] Stewart, John W. III. *BGP4: Inter-Domain Routing in the Internet.* Reading, MA: Addison-Wesley, 1999.

[SX 02] Stewart, Randall R., and Xie, Qiaobing. *Stream Control Transmission Protocol (STCP).* Reading, MA: Addison-Wesley, 2002.

[Ste *et al.* 04] Stevens, W. Richard, Fenner, Bill, and Rudoff, Andrew, M. *UNIX Network Programming: The Sockets Networking API.* Reading, MA: Addison-Wesley, 2004.

[Sti 06] Stinson, D. *Cryptography: Theory and Practice.* New York: Chapman & Hall/CRC, 2006.

[SW 05] Steinmetz, R., and Wehrle, K. *Peer-to-Peer Systems and Applications.* Berlin: Springer-Verlag, 2005.

[Tan 03] Tanenbaum, Andrew S. *Computer Networks,* 4th ed. Upper Saddle River, NJ: Prentice Hall, 2003.

[Tho 00] Thomas, S. *SSL and TLS Essentials.* New York: John Wiley & Sons, 2000.

[WV 00] Warland, J., and Varaiya, P. *High Performance Communication Networks.* San Francisco, CA: Morgan, Kaufmans, 2000.

[WZ 01] Wittmann, R., and Zitterbart, M. *Multicast Communication.* San Francisco: Morgan Kaufmann, 2001.

[Zar 02] Zaragoza, R. *The Art of Error Correcting Coding.* Reading, MA: Addison-Wesley, 2002.

ÍNDICE

Numéricos

A

B

C

D

E

F

G

H

I

J

K

N

O

S

U

V

W

X

Y

Z